XVME-601

68000/68010 Processor Module

74601-001 B

© 1993 XYCOM, INC.

Printed in the United States of America
Part Number 74601-001 B

XYCOM
750 North Maple Road
Saline, Michigan 48176-1292
734-429-4971 (phone)
734-429-1010 (fax)

XYCOM REVISION RECORD

Revision	*Description*	*Date*
A	Manual Released	12/86
B	Incorporated PCN 120	06/93

Copyright Information

This document is copyrighted by Xycom Incorporated (Xycom) and shall not be reproduced or copied without expressed written permission from Xycom.

The information contained within this document is subject to change without notice. Xycom does not guarantee the accuracy of the information and makes no commitment toward keeping it up to date.

Address comments concerning this manual to:

xycom
Technical Publications Dept.
750 North Maple Road
Saline, Michigan 48176

Part Number: ***74601-001 B***

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
1	**MODULE DESCRIPTION**	
1.1	Introduction	1-1
1.2	Manual Structure	1-2
1.3	Module Operational Description	1-2
1.3.1	VMEbus Master Interface	1-4
1.3.2	The 68681 Dual Universal Asynchronous Receiver /Transmitter	1-4
1.3.3	Interrupts	1-5
1.3.4	Status LEDs and Control Switches	1-6
1.3.5	System Resource Functions	1-6
1.3.6	Local DRAM	1-6
1.4	XVME-601 Processor Module Specifications	1-7
2	**XVME-601 INSTALLATION**	
2.1	Introduction	2-1
2.2	Location of Components Relevant to Installation	2-1
2.3	Jumpers	2-3
2.4	Jumper Descriptions	2-3
2.4.1	RESET/ABORT Switch Enable Jumpers	2-4
2.4.2	System Resource Function Jumpers	2-5
2.4.3	EPROM/RAM Type Selection Jumpers	2-7
2.4.4	VMEbus Interrupt Level Selection Jumpers	2-7
2.4.5	Bus Grant and Bus Request Level Selection Jumpers	2-8
2.5	Installing EPROM on the XVME-601 Processor Module	2-8
2.6	External Connector JK1	2-11
2.7	Module Installation	2-12
2.8	Installing the XVME-990/2 Monitor/RAM Kit	2-14
2.9	Installing a 6U Front Panel Kit	2-16
3	**PROGRAMMING**	
3.1	Introduction	3-1
3.2	The XVME-601 Processor Module Memory Map	3-1
3.2.1	The 68681 DUART Memory Map	3-3
3.3	Using the 68681 I/O Ports	3-5
3.4	68681 DUART Initialization	3-7
3.5	Interrupts	3-8

TABLE OF CONTENTS continued

CHAPTER	TITLE	PAGE
4	**XVME-600MON**	
4.1	Introduction	4-1
4.2	Referenced Documents	4-1
4.3	System Configuration	4-2
4.3.1	600MON Memory Map	4-3
4.3.1.1	Exception Vectors	4-5
4.3.2	600Mon I/O	4-5
4.4	Using 600MON	4-5
4.4.1	Expression Parameters	4-6
4.4.2	Address Parameters	4-7
4.4.3	Offset Registers	4-8
4.4.4	Data Parameters	4-9
4.5	Command Verification	4-10
4.6	Command Set	4-11
4.6.1	Display/Set Register	4-12
4.6.2	Block Fill	4-13
4.6.3	Block Initialize	4-14
4.6.4	Block Move	4-15
4.6.5	Breakpoint Set and Remove	4-16
4.6.6	Block of Memory Search	4-18
4.6.7	Block of Memory Test	4-19
4.6.8	Data Conversion	4-20
4.6.9	Display Formatted Registers	4-21
4.6.10	Dump Memory (S-Records)	4-22
4.6.11	Execute Programs	4-23
4.6.12	Go Direct Execute Program	4-25
4.6.13	Go Until Breakpoint	4-26
4.6.14	Help	4-27
4.6.15	Load (S-Records)	4-28
4.6.16	Memory Display/Disassemble	4-29
4.6.17	Memory Modify/Disassemble Assemble	4-31
4.6.18	Memory Set	4-33
4.6.19	Display Offsets	4-34
4.6.20	Port Attach and Detach	4-35
4.6.21	Port Format	4-36
4.6.22	Register Modify	4-37
4.6.23	Transparent Mode	4-38
4.6.24	Trace	4-39
4.6.25	Trace to Temporary Breakpoint	4-41
4.6.26	Verify (S-Records)	4-42
4.7	Using 600MON I/O Routines Through Trap 15	4-43

TABLE OF CONTENTS continued

CHAPTER	TITLE	PAGE
	APPENDICES	
A	VMEbus Connector/Pin Description	
B	Block Diagram, Assembly Drawing, and Schematics	
C	Quick Reference Guide	

LIST OF FIGURES

FIGURE	TITLE	PAGE
1-1	Module Operational Block Diagram	1-3
2-1	Location of Jumpers, Sockets, and Connectors	2-2
2-2	XVME-601 Memory Map (Factory-Shipped Configuration)	2-10
2-3	Connector JK1	2-11
2-4	Memory Map after Installing XVME-990 Monitor/ RAM Kit	2-15
2-5	Installation of an XVME-943 Front Panel	2-17
3-1	XVME-601 Processor Module Memory Map	3-2
3-2	68681 DUART Memory Map	3-4
4-1	Typical XVME-601 System Configuration	4-2
4-2	600MON Memory Map	4-4

LIST OF TABLES

TABLE	TITLE	PAGE
1-1	XVME-601 Processor Module Specifications	1-7
2-1	Jumper List	2-3
2-2	RESET/ABORT	2-4
2-3	System Resource Function Jumpers	2-6
2-4	EPROM Selection Jumpers for Sockets U30 & U29	2-7
2-5	Interrupt Selection Jumpers	2-6
2-6	Bus Grant and Bus Request Level Selection Jumpers	2-8
2-7	"Local" RAM/PROM Memory Configurations	2-9
2-8	Connector JK1 Pin Designations	2-12
3-1	68681 Parallel Input Status Indications	3-5
3-2	68681 Output Control Functions	3-6

LIST OF TABLES continued

TABLE	TITLE	PAGE
3-3	XVME-601 Processor Module Interrupts	3-8
4-1	Offset Examples	4-8
4-2	600MON Commands	4-11
4-3	Available User Routines	4-44
4-4	Registers Used by Function	4-45

Chapter 1

MODULE DESCRIPTION

1.1 INTRODUCTION

The XVME-601 Processor Module is a single-high, VMEbus compatible board, which uses a 68000 or 68010 CPU running at 10 MHz. The module contains two byte-wide memory sockets for EPROM devices. These sockets allow a maximum of 128K bytes of EPROM to be installed. On board DRAM chips provide 512K bytes of local RAM. The CPU is capable of accessing the local DRAM and EPROM without the need for "wait states". XYCOM's XVME-990/2 600MON Debugger Kit is available for the XVME-601.

The XVME-601 Processor Module provides two RS-232 serial ports, and a 16-bit programmable timer via an on-board 68681 Dual Asynchronous Receiver/Transmitter (DUART). The module also incorporates three front panel LEDs to indicate diagnostic PASS/FAIL status (diagnostics are included in the 601 monitor), as well as CPU HALT status.

In addition, the XVME-601 Processor Module provides all of the VMEbus utilities required for a complete system, including:

- SYSCLK
- SYSRESET
- A single level arbiter
- A Bus Timer
- IACK Daisy Chain Driver

The XVME-601 Processor is specified as an A24/A16:D16/D08 (EO) VMEbus Master, and as an IH(0)-IH(7) interrupt handler.

1.2 MANUAL STRUCTURE

It is the aim of this first chapter to introduce the user to the general specifications and functional capabilities of the XVME-601 Processor Module. Successive chapters will develop the various aspects of module specification and operation in the following progression:

Chapter One - A general description of the XVME-601 Module, including complete functional and environmental specifications, VMEbus compliance information, and a detailed block diagram.

Chapter Two - Module installation information, covering jumpers and connector pinouts.

Chapter Three - Includes information on the module memory map, the 68681 DUART (including a DUART initialization program), and the interrupt structure.

Chapter Four - Contains information on the operation and use of the optional 600MON debug monitor.

The appendices at the rear of this manual are designed to introduce and reinforce a variety of module related topics including:backplane signal/pin descriptions, a block diagram and schematics, and a quick reference section.

NOTE

In order to fully document the complex versatility of the XVME-601 and the 68681 DUART device, a manual kit is being shipped with the XVME-601 Module (the manual kit is referenced as XYCOM Part #74601-001). This kit consists of two parts: a *Motorola 68681 Manual (referenced as XYCOM part # 74601-003), and this document, the XVME-601 Processor Module Manual (referenced as XYCOM Part #74601-002).

The 68681 Manual covers all information on the operation lay-out, and programming of the DUART. The XVME-601 Manual covers module hardware specifics, register access addresses, and operational programming constraints.

* MC68681 Dual Asynchronous Receiver/Transmitter Manual, Motorola Inc. 1985

1.3 MODULE OPERATIONAL DESCRIPTION

Figure 1-1 shows an operational block diagram of the XVME-601 Processor Module.

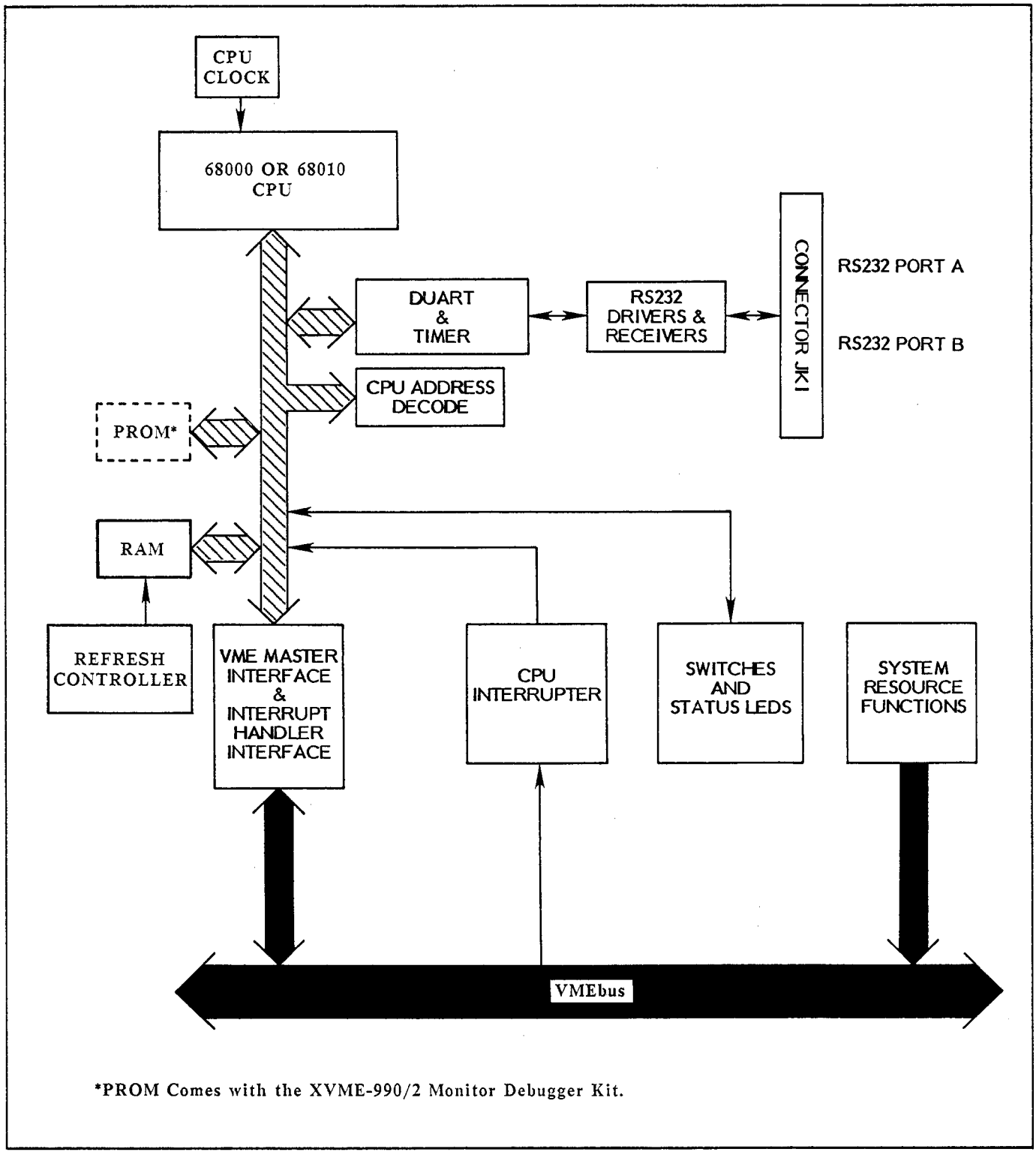


Figure 1-1. Module Operational Block Diagram

1.3.1 VMEbus Master Interface

The VME master interface on the XVME-601 Processor Module supports the following bus cycles:

- A24 (Address Modifier Codes 39H, 3AH, 3DH, or 3EH)
- A16 (Address Modifier Codes 29H, 2DH)
- D16
- D8 (Even or Odd Transfers)
- Read-Modify-Write (RMW) Cycles - D8 (Even or Odd Transfers)
- Interrupt Acknowledge Cycles - D8 (Odd Transfers Only)

Note that RMW cycles can use either even or odd byte transfers, while IACK cycles can use only odd byte transfers. The XVME-601 Module can be configured (Via Jumpers) to use any of the four bus arbitration levels (see Chapter 2).

The VMEbus is requested when the CPU executes an off-board memory or IACK cycle, and the XVME-601 Module does not presently have control of the bus. After requesting the bus, wait cycles will be inserted until the bus is acquired and the slave response with DTACK* or BERR*. When the module has control of the VMEbus, the wait states due to the acquisition phase will cease.

The bus release mechanism may be Release When Done (RWD), Release On Request (ROR), release on BCLR* (ROBC), or release on ACFAIL* (ROACF). In the last case an interrupt can be generated when ACFAIL* is asserted, and the CPU can release the bus if desired. Early BBSY* release is supported when the XVME-601 Module decides to release the bus during the last master cycle.

1.3.2 The 68681 Dual Universal Asynchronous Receiver Transmitter (DUART)

A 68681 DUART is used to provide two RS-232 serial communication channels, a 16-bit timer, and two ports dedicated to module status and control I/O. The two asynchronous serial channels (labeled Channel A and Channel B are configured as RS-232 Data Communication Equipment (DCE) channels. Independent baud rate generators allow most of the popular data rates from 75 to 19.2K baud. In addition to the standard receive and transmit lines, Channel A has a modem control input (RTS) and output (CTS). All serial communication lines are accessible at the 26 pin connector (labeled JK1) located on the module front panel (refer to Chapter 2 for the Channel A and B pinouts).

The 68681 contains a timer/counter which is independent from the serial channel baud rate generators. The timer can be programmed from the serial channel baud rate generators. The timer can be programmed from the serial channel baud rate generators. The timer is set up to use 3.6864 MHz/16 or 230.4 KHz as its time base. This will allow periodic interrupts or square wave outputs ranging from 17.4uS to 569mS (refer to the MC68681 Dual Asynchronous Receiver/Transmitter Manual for information on programming the timer).

The 68681 DUART on the XVME-601 Processor Module is configured to provide 6 parallel inputs and 8 parallel outputs. These I/O points are used by the CPU to control and monitor a variety of module functions. The input lines are used by the CPU to monitor the following:

- the RTS line on Channel A
- when the module has control of the VMEbus
- SYSFAIL*, BCLR*, and ACFAIL* lines
- the ABORT button on the front panel

The output lines are used by the CPU to control the following functions:

- the Channel A CTS output
- the type of bus release mechanism to be used by the module
- the PASS and FAIL LEDs on the module front panel
- clearing and enabling interrupts

1.3.3 Interrupts

The Processor on the XVME-601 Processor Module can be interrupted via all seven VMEbus interrupts, plus 2 local and 2 special interrupt sources. The local interrupt sources include the ABORT push-button, and the 68681 DUART, and the special interrupts include ACFAIL* and SYSFAIL*. Note that ACFAIL* and SYSFAIL* are considered special interrupts in this case because they must be individually enabled by the CPU through the 68681 outputs. Refer to 68681 Manual for information on the types of interrupts which the DUART itself can generate.

One of the outputs on the 68681 DUART is used to enable/disable all interrupt capability, and some are used to individually enable/clear local and special interrupt capabilities. Refer to Section 3.3 for information on controlling the 68681 outputs.

The local interrupts have priority over the VMEbus interrupts when both exist on the same level.

1.3.4 Status LEDs and Control Switches

Two momentary push-button switches can be found on the front panel of the module, they are labeled RESET and ABORT. RESET will reset the module and the VMEbus (via SYSRESET*). The DRAM refresh logic is not disabled during reset, therefore the module may be reset without effecting the contents of the DRAM. ABORT generates a level 7 interrupt to the CPU.

In addition, three LEDs can be found on the module front panel. The LEDs are labeled FAIL, PASS, and HALT. Fail is on when the CPU is asserting SYSFAIL*. The PASS light is controlled directly by the CPU. HALT is on when the CPU enters halt state.

1.3.5 System Resource Functions

The following system resource functions exist on the XVME-601 Processor Module:

- SYSCLK driver
- SYSRESET driver
- IACK daisy-chain driver
- Single-level arbiter
- Bus timeout

The system resource functions can be enabled/disabled via jumpers (refer to Chapter 2). The Module is shipped from the factory with the system resource functions enabled.

1.3.6 Local DRAM

On board DRAM provides 512K of random access memory. No wait states are required to access the DRAM except when the CPU tries to access the DRAM during a refresh cycle. A refresh cycle requires 3 clock periods (300nS). Therefore the maximum number of wait states will be 3. No wait states will be encountered during a refresh cycle when the CPU accesses anything other than the DRAM. The ram exists from address 000000H through 07FFFFH. The refresh circuitry is not disabled during the reset, therefore the module may be reset with out effecting the contents.

1.4 XVME-601 PROCESSOR MODULE SPECIFICATIONS

Table 1-1 is a list of the operational and environmental specifications for the XVME-601 Processor Module.

Table 1-1. XVME-601 Processor Module Specifications

Characteristic	Specifications
Processor	68000 or 68010
Speed	10 MHz
Memory Capacity	
EPROM	2 sockets or up to 128K bytes
DRAM	512K bytes
Serial Ports	
Number	2
Compatibility	RS-232
Baud Rates	75 - 19.2K baud, programmable
Signals	Port A - TxD, RxD, RTS, CTS Port B - TxD, RxD
Time Base Generator	17.4uSec to 569mSec.
Front Panel Indicators	FAIL (red), PASS (green), and HALT (red)
Front Panel Switches	RESET and ABORT
Power Requirements	+5 volts @ 2.8A Max. (2.5 A Typ.) +/- 12 volts @ 100mA Max. (10mA Typ.)
Temperature	
Operating	0 to 65 C (32 to 149 F)
Non-operating	-40 to 85 C (-40 to 158 F)
Humidity	5 to 95% RH non-condensing (Extremely low humidity may require protection against static discharge.)
Altitude	
Operating	Sea-level to 10,000 ft. (3048m)
Non-operating	Sea-level to 50,000 ft. (15240m)

Table 1-1. XVME-601 Processor Module Specifications cont.

Characteristic	Specifications
Vibration	
Operating	5 to 2000 Hz .015 in. peak-to-peak 2.5g max.
Non-operating	5 to 2000 Hz .030 in. peak-to-peak 5.0g max.
Shock	
Operating	30 g peak acceleration 11 mSec duration
Non-operating	50 g peak acceleration 11 mSec duration
VMEbus Compliance	• A24/A16:D16/D08(EO) DTB Master • RMW Capability • IH(1)-IH(7) D08(O) Interrupt Handler • SGL Arbiter • R(0-3) Bus Requester • RWD, ROR, or ROBC (DYN) bus release • ROACF (software controlled) bus release • BTO (16,32,64,128)STAT • IDCD IACK Daisy Chain driver • SYSCLK and SYSRESET driver • Monitors SYSFAIL, ACFAIL, and BCLR • SINGLE - 3U Form Factor

Chapter 2

XVME-601 INSTALLATION

2.1 INTRODUCTION

This chapter provides the information needed to configure and install the XVME-601 Processor Module.

2.2 LOCATION OF COMPONENTS RELEVANT TO INSTALLATION

The jumpers, PROM sockets, and connectors on the XVME-601 Processor Module are illustrated in Figure 2-1.

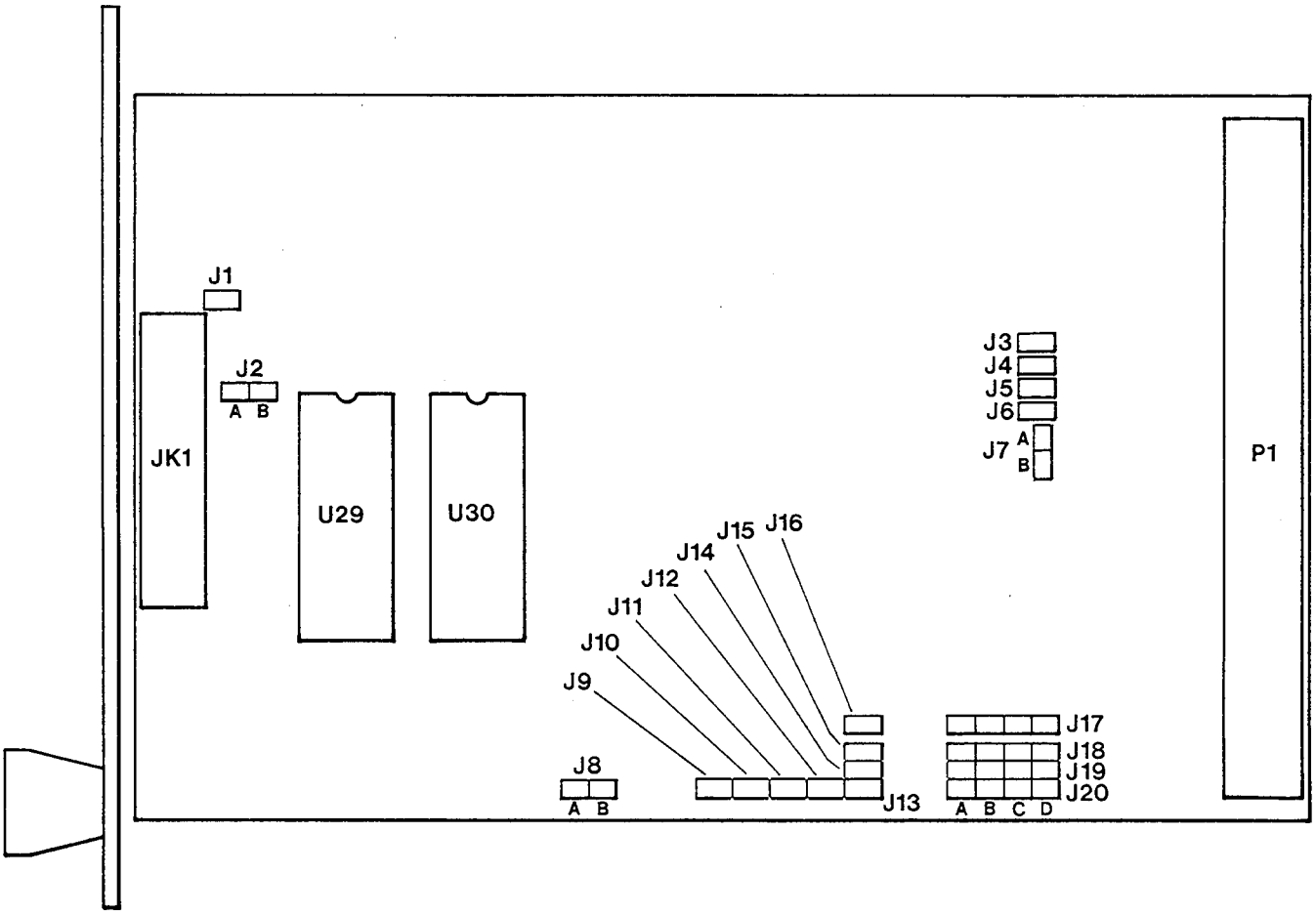


Figure 2-1. Location of Jumpers, Sockets, and Connectors.

2.3 JUMPERS

Prior to installing the XVME-601 Processor Module, it will be necessary to configure several jumper options. The configuration of the jumpers is dependant upon which of the module capabilities are required for a given application, and the type of PROM used for "local" memory. The following table shows a list of the jumpers and a brief description of their use.

Table 2-1. Jumper List

XVME-601 Jumpers

Jumper	Use
J1	This jumper is used to enable/disable the reset switch on the front panel of the module.
J3,J4,J5,J6,J7, and J17	These jumpers are used to enable/disable the system resource functions
J2,J8	These jumpers are configured to match the type and size of EPROM which is installed on the XVME-601.
J9,10,11,12,13,14,15	These jumpers are used to determine which VMEbus interrupt levels (1-7) can be used to interrupt the CPU.
J16	This jumper enables/disables the abort switch on the front panel of the module.
J18,J19, & J20	These jumpers are used to select the bus request and bus grant levels.

2.4 JUMPER DESCRIPTIONS

Each of the following subsections examines the jumper options in closer detail, showing specifically when and how jumpers should be configured.

2.4.1 RESET/ABORT Switch Enable Jumpers (J1 & J16)

There are two momentary push-button switches which can be found on the front panel of the module. The switches are labeled RESET and ABORT. When the RESET button is enabled (by installing jumper J1), it can be used to reset the CPU and the VMEbus. When the ABORT button is enabled (by installing jumper J16), it can be used to generate a level 7 interrupt to the CPU. Table 2-2 shows the configuration of these jumpers for enabling/disabling the front panel switches.

Table 2-2. RESET/ABORT Switch Enable Jumpers

Jumper J1	Jumper J16	RESET Switch	ABORT Switch
In	-	Enabled	-
Out	-	Disabled	-
-	In	-	Enabled
-	Out	-	Disabled

The XVME-601 Module is configured at the factory with both RESET and ABORT activated.

NOTE

Use of the RESET button activates the SYSRESET* line if jumpers J1 and J6 are installed, thereby resetting the entire backplane. The RESET button cannot be used to just reset the XVME-601 board.

2.4.2 System Resource Function Jumpers (J3,J4,J5,J6,J7)

The XVME-601 Processor Module provides all of the VMEbus utilities required for a complete system, including:

- SYSCLK
- SYSRESET
- A single level arbiter
- A Bus Timer

NOTE

For the most part, the system resource functions are employed as a group (i.e. either they are all enabled, or they are all disabled). However, the SYSRESET function can be used by itself as long as jumpers J1 and J6 are installed.

The following are the definitions of the system resource jumpers:

Table 2-3. System Resource Function Jumpers

Jumper	Definition
J3	This jumper enables/disables the SYSCLK driver.
J4	This jumper enables/disables the generation of BERR* following the bus timeout function.
J5	This jumper enables/disables the single level arbiter.
J6	This jumper enables/disables the SYSRESET line.
J17A	This jumper sets a bus timeout duration of 16uS.
J17B	This jumper sets a bus timeout duration of 32uS.
J17C	This jumper sets a bus timeout duration of 64uS.
J17D	This jumper sets a bus timeout duration of 128uS.
J7A	This jumper enables the IACK daisy chain driver.
J7B	To completely disable the on-board system resource functions, J3, J4, J5, J6, & J7A must be removed, and J7B must be installed.

NOTE

In order to fully implement the system resource functions, the Bus Grant In/Out and Bus Request Lines must be set to Level 3 (Jumpers J18, J19, and J20 set in the D position.

By installing J3, J4, J5, J6, J17(A, B, C, or D), J18D, J19D, J20D and J7A the system resource functions are enabled. By removing J3, J4, J5, J6, and J7A, and installing J7B, the system resource functions are disabled. The XVME-601 is configured at the factory with the system resource functions activated, and the bus timeout duration set for 16uS.

2.4.3 EPROM Type Selection Jumpers

There are two 28-pin sockets provided on the module for ROM devices. Sockets U29 and U30 are dedicated for use by the EPROM devices, and they can accept 2764, 27128, 27256, or 27512 EPROMs (200 nS access). The EPROMs are installed at U29 and U30 must both be the same type of EPROM. Jumpers J8 and J2 are used to select the type of EPROM to be installed at the U29 and U30 sockets. Table 2-4 shows how the jumpers should be configured for the various EPROM possibilities.

Table 2-4. EPROM Selection Jumpers for Sockets U29 and U30

J8	J2	Device Selected
B	B	2764
B	B	27128 (factory shipped config.)
B	A	27256
A	A	27512

2.4.4 VMEbus Interrupt Level Selection Jumpers

All seven VMEbus interrupts are recognized by the XVME-601 Module. Table 2-5 shows which jumpers (when installed) will enable the various interrupt levels.

Table 2-5. Interrupt Selection Jumpers

Jumper	Interrupt Level and Signal Name
J9	LEVEL 1 IRQ1*
J10	LEVEL 2 IRQ2*
J11	LEVEL 3 IRQ3*
J12	LEVEL 4 IRQ4*
J13	LEVEL 5 IRQ5*
J14	LEVEL 6 IRQ6*
J15	LEVEL 7 IRQ7*

NOTE

The XVME-601 is configured at the factory with all interrupt selection jumpers installed.

In addition, there are two local sources and two special sources which can interrupt the CPU. Refer to Section 3.5 of this Manual for additional information on the local and special interrupt sources.

2.4.5 Bus Grant and Bus Request Level Selection Jumpers

Jumpers J18, J19, and J20 are used to select the bus request and bus grant levels as shown in Table 2-6.

Table 2-6. Bus Grant and Bus Request Level Selection Jumpers

Jumper Position	Jumper J18 Bus Grant In	Jumper J19 Bus Grant Out	Jumper J20 Bus Request
A	BG0IN	BG0OUT	BR0
B	BG1IN	BG1OUT	BR1
C	BG2IN	BG2OUT	BR2
*D	BG3IN	BG3OUT	BR3

*Factory Shipped Position

NOTE:

For a given configuration, J18, J19, and J20 must all be in the same position. Position "D" is required when using the on-board system resource functions. In order to pass the unselected Bus Grant signals around the XVME-601, the backplane jumpers should be used.

2.5 INSTALLING EPROM ON THE XVME-601 PROCESSOR MODULE

As was previously mentioned, there are a total of two 28-pin sockets for use by EPROM devices on the XVME-601 Module.

Installing EPROMs in sockets U29 and U30 is simply a matter of setting the jumpers to match the devices as shown in Table 2-4, referencing the notched ends of the chips as shown in Figure 2-1, and installing the devices. The EPROM installed in socket U30 should contain the even byte addresses and the EPROM installed in socket U29 should contain the odd byte addresses.

Table 2-7 shows how various devices can be employed to arrive at different "local" memory configurations and Figure 2-2 shows the XVME-601 Memory Map.

Table 2-7. "Local" PROM Memory Configurations

Type of Device	# of Devices Installed	Overall Memory Size Provided
2764 EPROM	2	16K Bytes
27128 EPROM	2	32K Bytes
27256 EPROM	2	64K Bytes
27512 EPROM	2	128K Bytes

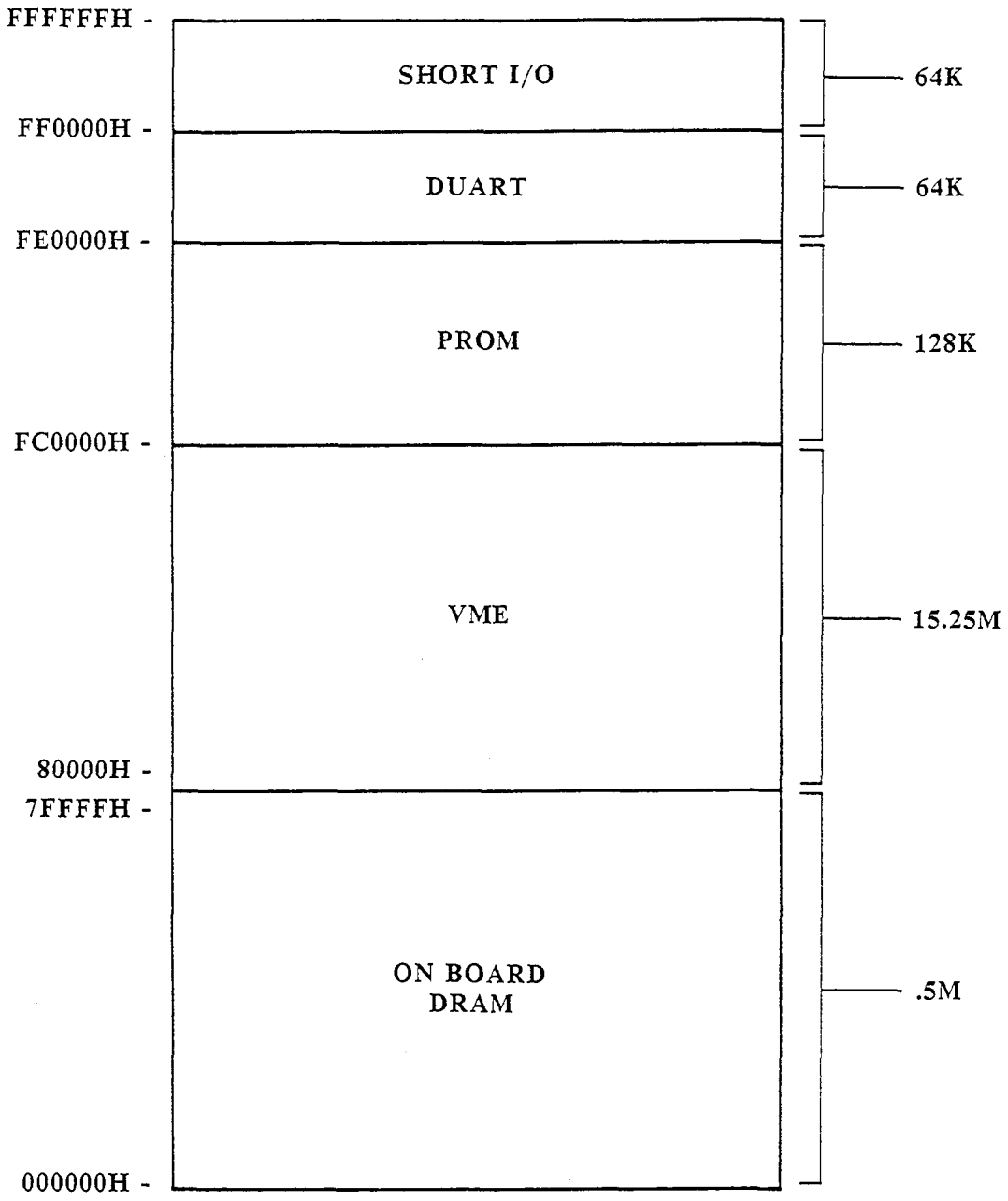


Figure 2-2. XVME-601 Memory Map (Factory-Shipped Configuration)

NOTE

When devices totaling less than 128K Bytes of memory are installed in the PROM sites, the unused portion of the 128K becomes a "shadow-image" of the space occupied by the installed devices. For example, if two 2764 EPROMs are installed in the PROM sockets they will only occupy 16K bytes of the 128K space mapped out for them. Thus, the original 16K bytes will be "shadowed" 8 times, thereby filling the 128K PROM map.

2.6 EXTERNAL CONNECTOR JK1

The XVME-601 Processor Module provides two asynchronous serial channels (A and B) which are configured as RS-232 "DCE" equipment. Both channels have the traditional transmit (TxD) and receive (RxD) lines, and in addition channel A has a modem control input (RTS) and a modem control output (CTS). The RS-232 signals are accessible via a 26-pin connector (JK1) located on the module front panel. Figure 2-3 shows the module front panel and how the pins are situated in the connector.

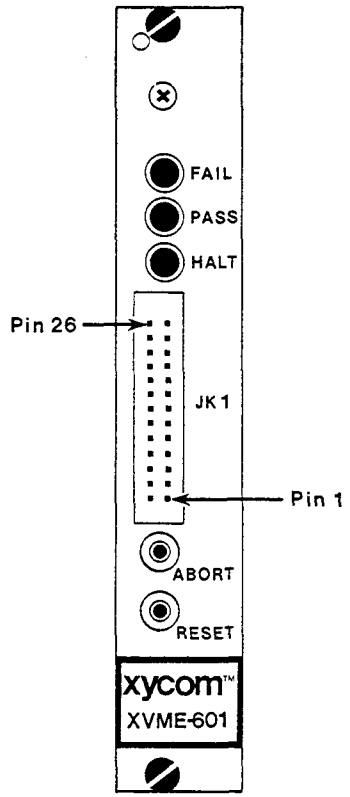


Figure 2-3. Connector JK1

Table 2-8 shows the module front panel and the pin designations for connector JK1.

Table 2-8. Connector JK1 Pin Designations

Pin #	Pin Designation	Signal Direction
1	NC	-
2	NC	-
3	TxD Channel A	Input
4	NC	-
5	RxD Channel A	Output
6	NC	-
7	RTS Channel A	Input
8	NC	-
9	CTS Channel A	Output
10	NC	-
11	NC	-
12	NC	-
13	GND Channel A	-
14	TxD Channel B	Input
15	NC	-
16	RxD Channel B	Output
17	NC	-
18	NC	-
19	NC	-
20	NC	-
21	NC	-
22	NC	-
23	NC	-
24	GND Channel B	-
25	NC	-
26	NC	-

2.7 MODULE INSTALLATION

XYCOM XVME modules are designed to comply with all physical and electrical VMEbus backplane specifications. The XVME-601 Processor Module is a single-high VMEbus module, and as such, only requires the P1 backplane.

CAUTION

Never attempt to install or remove any boards before turning off the power to the bus, and all related external power supplies.

Prior to installing a module, you should determine and verify all relevant jumper configurations, and all connections to external devices or power supplies. (Please check the jumper configuration against the diagrams and lists in this manual.)

To install a board in the cardcage, perform the following steps:

1) Make certain that the particular cardcage slot which you are going to use is clear and accessible.

2) Center the board on the plastic guides in the slot so that the handle on the front panel is towards the bottom of the cardcage.

3) Push the card slowly toward the rear of the chassis until the connectors engage (the card should slide freely in the plastic guides).

4) Apply straight-forward pressure to the handle located on the front panel of the module until the connector is fully engaged and properly seated.

NOTE

It should not be necessary to use excessive pressure or force to engage the connectors. If the board does not properly connect with the backplane, remove the module and inspect all connectors and guide slots for possible damage or obstructions.

2.8 INSTALLING THE XVME-990/2 MONITOR KIT (OPTIONAL)

When installed, the XVME-990/2 Monitor kit provides the XVME-601 Module with two 16K byte EPROMs (27128), contains XYCOM's 600MON debug Monitor.

Each of the two EPROM devices will be labeled on the top. In the lower right corner of each label will be either a "0" or a "1". The device labeled "0" must be inserted in PROM socket U30 and the device labeled "1" must be inserted in PROM socket U29. Again, make sure the notched ends of the chips are positioned as shown in Figure 2-1.

Figure 2-4 shows the Memory Map after the XVME-990/2 PROM has been installed.

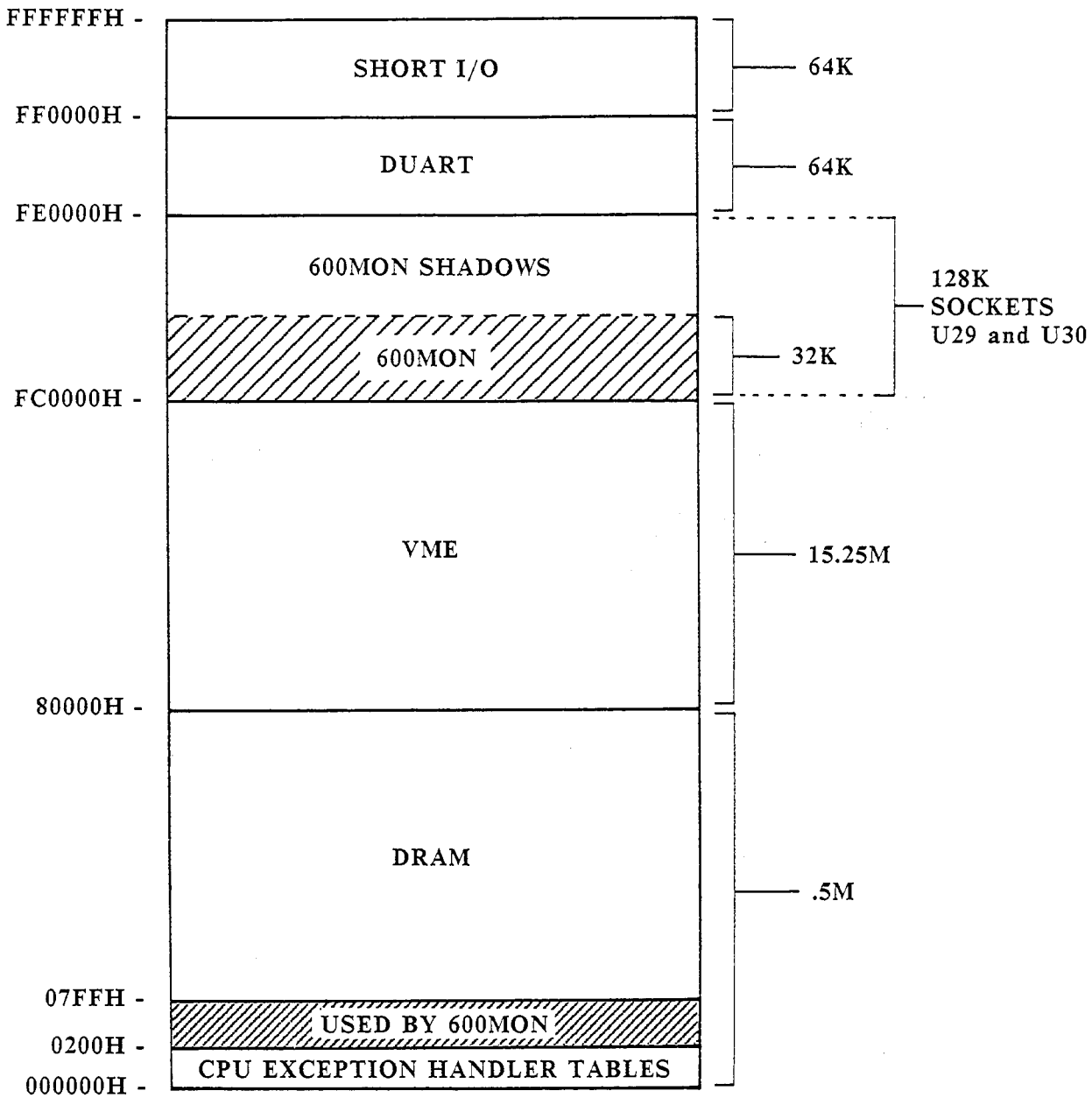


Figure 2-4. Memory Map After Installing XVME-990/2 Monitor Kit

2.9 INSTALLING A 6U FRONT PANEL KIT (OPTIONAL)

XYCOM Model Number XVME-943 is an optional 6U front panel kit designed to replace the existing 3U front panel on the XVME-601. The 6U front panel facilitates the secure installation of single-high modules in those chassis which are designed to accommodate double-high modules. The following is a step-by-step procedure for installing the 6U front panel on an XVME-601 Module (refer to Figure 2-5 for a graphic depiction of the installation procedure).

1) Disconnect the module from the bus.

2) Remove the screw and plastic collar assemblies (labeled #6 and #7) from the extreme top and bottom of the existing 3U front panel (#11), and install the screw assemblies in their corresponding locations on the 6U front panel.

3) Slide the module identification plate (labeled #13) from the handle (#9) on the 3U front panel. By removing the screw/nut found inside the handle, the entire handle assembly will separate from the 3U front panel. Remove the counter-sunk screw labeled #8 to separate the 3U front panel from the printed circuit board (#12).

4) Line-up the plastic support brackets on the printed circuit board with the corresponding holes in the 6U front panel (i.e. the holes at the top and top-center of the panel). Install the counter-sunk screw (#8) in the hole near the top-center of the 6U panel, securing it to the lower support bracket on the printed circuit board.

5) Install the handle assembly (which was taken from the 3U panel) at the top of the 6U panel, using the screw and nut previously attached inside the handle. After securing the top handle, slide the module identification plate in place.

6) Finally, install the bottom handle (i.e. the handle that accompanies the kit - labeled #2) using the screw and nut (#3 & #5) provided. Slide the XYCOM VMEbus I.D. plate (#4) in place on the bottom handle. The module is now ready to be re-installed in the backplane.

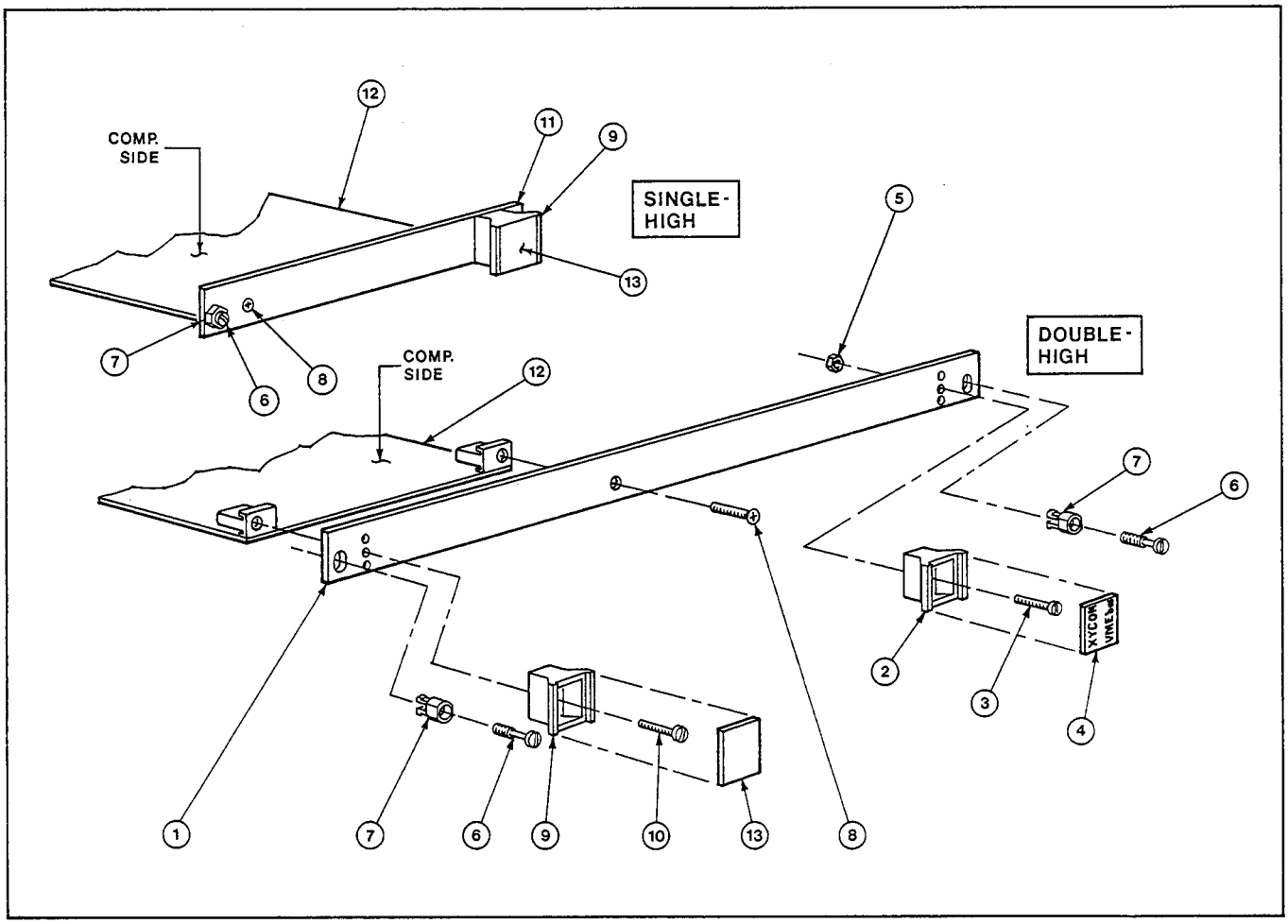


Figure 2-5. Installation of an XVME-943 6U Front Panel

Chapter 3

PROGRAMMING

3.1 INTRODUCTION

This Chapter will briefly examine the overall module memory map, the 68681 DUART memory map, the dedicated I/O points on the 68681 DUART, interrupts, and initialization of the 68681 DUART. In order to demonstrate the correct initialization sequence for the 68681 DUART a sample initialization routine (with comments) has been incorporated in this chapter. For a complete explanation on how to program and maximize the functionality of the 68681 DUART, refer to the accompanying 68681 Manual.

3.2 THE XVME-601 PROCESSOR MODULE MEMORY MAP

Figure 3-1 shows the XVME-601 Module memory map as it would appear with the factory configuration.

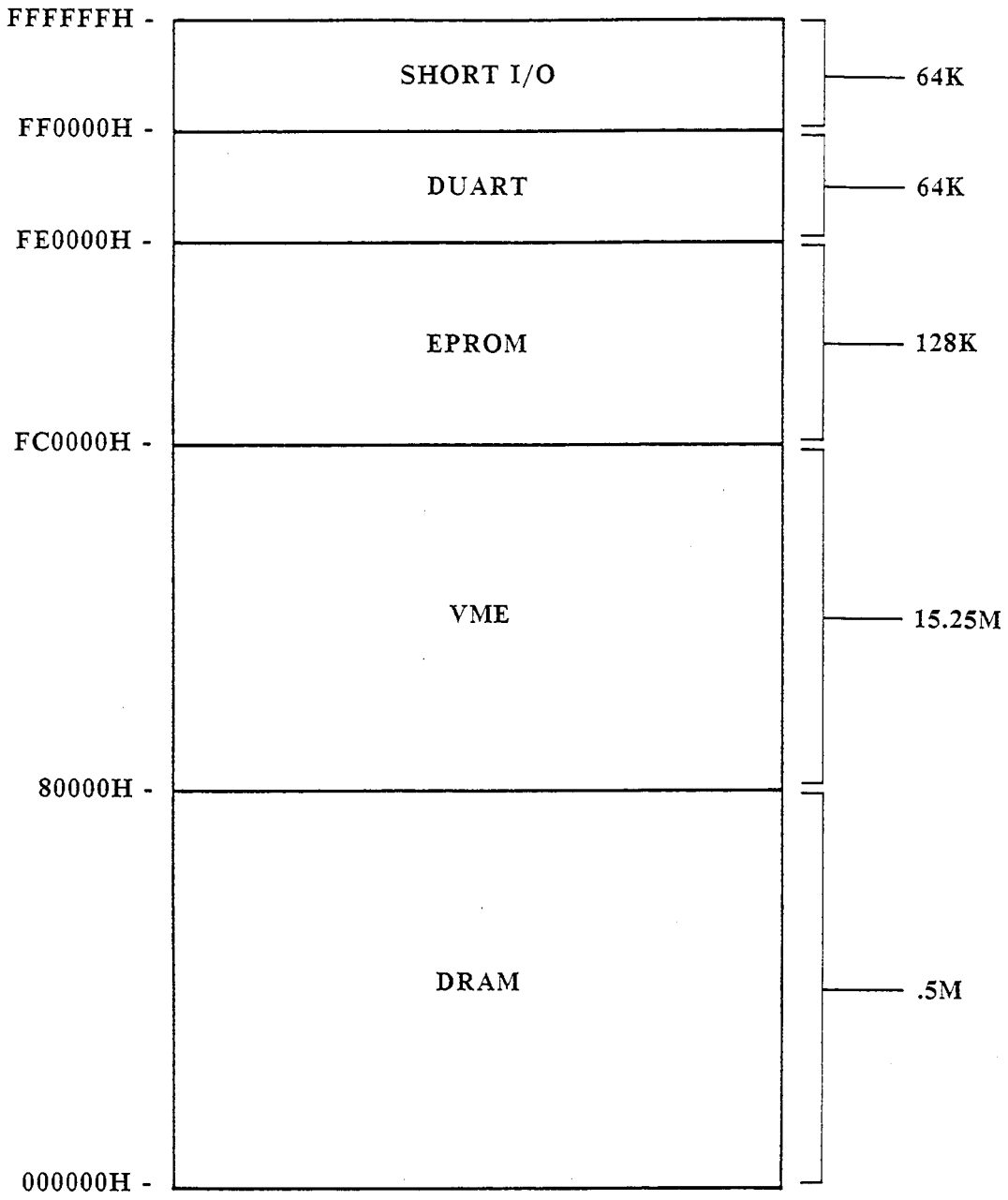


Figure 3-1. XVME-601 Processor Module Memory Map

NOTE

When the CPU starts running after reset the first four bus cycles will be read to acquire the initial Supervisor Stack Pointer (SSP) and Program Counter (PC). Hardware will map these four bus cycles into address FC0000H for the SSP and address FC0004 for the PC. This convention allows the exception vectors to reside in DRAM.

The area from FC0000H to FDFFFFH is dedicated for the optional installation of 2 ROM devices (XYCOM's 600MON EPROMs for example). If the ROM devices installed do not fully occupy the entire 128K of address space mapped out, then the remaining space will contain shadow images of the occupied area. ROM must be installed at U29 and U30 due to SSP and PC acquisition during power up.

The area from FE0000H to FEFFFFH is mapped for the various 68681 DUART control, command, and data registers (refer to section 3.2).

3.2 THE 68681 DUART MEMORY MAP

As mentioned in the previous Section, the 68681 DUART occupies the area of the XVME-601 memory map from FE0000H to FEFFFFH. However, only 16 of the total 64K bytes in the mapped area are used for the DUART. These 16 bytes go from FE0001H to FE001FH (odd bytes only). Most of the 16 byte locations pertaining to the DUART have dual definitions. That is, during a WRITE cycle a register will have one definition, and during a READ cycle the same register will have another Definition (Refer to Figure 3-2 or the Programming and Register Description Section of the MC68681 Manual). Figure 3-2 shows the memory map for the 68681 and the definition of the registers during both a READ cycle and a WRITE cycle.

CAUTION

The 16 bytes used for the 68681 DUART registers reside on the odd bytes only! Any byte accesses to the even byte at these locations will not return DTACK*, and the processor will "hang up" until it is reset. However, word accesses are permitted.

ADDRESS	READ	WRITE
FE0001H	Mode Register A (MR1A,MR2A)	Mode Register A (MR1A,MR2A)
FE0003H	Status Register A (SRA)	Clock Select Register A (CSRA)
FE0005H	Do Not Access *	Command Register A (CRA)
FE0007H	Receiver Buffer A (RBA)	Transmitter Buffer A (TBA)
FE0009H	Input Port Change Reg. (IPCR)	Aux. Control Register (ACR)
FE000BH	Interrupt Status Reg. (ISR)	Interrupt Mask Register (IMR)
FE000DH	Current Mode: Current MSB of Counter (CUR)	Counter/Timer Upper Register (CTUR)
FE000FH	Counter Mode: Current LSB of Counter (CLR)	Counter/Timer Lower Register (CTLR)
FE0011H	Mode Register B (MR1B,MR2B)	Mode Register B (MR1B,MR2B)
FE0013H	Status Register B (SRB)	Clock-Select Register B (CSRB)
FE0015H	Do Not Access*	Command Register B (CRB)
FE0017H	Receiver Buffer B (RBB)	Transmitter Buffer B (TBB)
FE0019H	Interrupt-Vector Register (IVR)	Interrupt-Vector Register (IVR)
FE001BH	Input Port (Unlatched)	Output Port Config. Reg. (OPCR)
FE001DH	Start Counter Command **	Output Port Register (OPR) — Bit Set Command**
FE001FH	Stop Counter Command **	Output Port Register (OPR) — Bit Reset Command**

*This address location is used for factory testing of the DUART and should not be read. Reading this location will result in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

** Address triggered commands

Figure 3-2. 68681 DUART Memory Map

3.3 USING THE 68681 I/O PORTS

The 68681 DUART has two multifunction I/O ports: one 6-bit parallel port for input (IP0-IP5), and one 8-bit parallel port for output (OP0-OP7). These I/O ports have been dedicated to specific monitor and control functions on the XVME-601 Module. The input lines (IP0-IP5) are used by the CPU to monitor certain module functions, and the output lines can be programmed (by using bit set and reset commands) to control certain module functions. Table 3-1 shows how the status of each of the six input lines on the 68681 DUART is interpreted on the XVME-601 Module.

Table 3-1. 68681 Parallel Input Status Indications

Input Line	Status	Indication
IP0	0	The Channel A RTS input is asserted
IP0	1	The Channel A RTS input is negated
**IP1	0	The XVME-601 has control of the VMEbus
**IP1	1	The XVME-601 does not have control of the Bus
IP2	0	SYSFAIL* is asserted
IP2	1	SYSFAIL* is negated
IP3	0	BCLR* is asserted
IP3	1	BCLR* is negated
IP4	0	ACFAIL* is asserted
IP4	1	ACFAIL* is negated
IP5	0	The ABORT button is pressed
IP5	1	The ABORT button is released

**Input line IP1 is used to coordinate changing the bus release control output lines OP1 and OP2--refer to Table 3-2.

Table 3-2 shows the functions that are controlled by setting or resetting the 68681 DUART output lines OP0-OP7. Setting the output lines entails writing a bit set command byte (1 equals set, 0 equals no change) to the bit set command register (location FE001DH). Output Bits are reset by writing a bit reset command byte (1 equals reset, 0 equals no change) to the bit reset command register (location FE001FH). Refer to the description of the Output Port Register in the MC68681 Manual for additional information on setting/resetting the output lines. All 68681 outputs will assume the RESET state when the module is reset, and when the CPU enters the HALT state.

Table 3-2. 68681 Output Control Functions

Output Line	Output Command	Function Performed
OP0	RESET	Negates the Channel A CTS Output
OP0	SET	Asserts the Channel A CTS Output
OP1		*SEE BELOW
OP2		
OP3	RESET	Asserts SYSFAIL* and Turns the FAIL LED On.
OP3	SET	Negates SYSFAIL* and Turns the FAIL LED Off.
OP4	RESET	Clears and disables interrupts generated by SYSFAIL*
OP4	SET	Enables SYSFAIL* Interrupts
OP5	RESET	Clears and disables interrupts generated by ACFAIL* or the ABORT Button.
OP5	SET	Enables Interrupts via the ABORT Button or ACFAIL*
OP6	RESET	Turns Off the PASS LED
OP6	SET	Turns On the PASS LED
OP7	RESET	Disables ALL Interrupts to the CPU
OP7	SET	Allows the CPU to be Interrupt

*OP1 and OP2 are used together to control the VMEbus release mechanism in the following fashion:

OP2	OP1	Bus Release Mechanism
RESET	RESET	Release When Done (RWD)
RESET	SET	Release On Request (ROR)
SET	RESET	Release On BCLR*
SET	SET	Never Release the Bus

NOTE

When changing the bus release mechanism, the following sequence must be followed:

1. RESET OP1 and OP2 by writing 06H to FE001FH
2. Monitor Input Line IP1 and wait for it to set to 1
3. Set desired bus release mechanism by writing to FE001DH

3.4 68681 DUART INITIALIZATION

The 68000 program which follows demonstrates a general method of initialization for the 68681 DUART on the XVME-601 Processor Module. The following are the equates which are used in the DUART initialization program:

```
MR1     EQU     $01         ;   Mode Register 1
MR2     EQU     $01         ;   Mode Register 2
CSR     EQU     $03         ;   Clock Select Register
STR     EQU     $03         ;   Status Register
CR      EQU     $05         ;   Command Register
THR     EQU     $07         ;   Transmit Holding Register
RHR     EQU     $07         ;   Receive Holding Register
```

68681 DUART Initialization Program

Object: To Initialize Channel A

```
A   00000000  207C00FE0000  MOVEA.L#$FE0000,A0   ;  This is Channel A
                                                     Address
A   00000006  117C00100005  MOVE.B #$10,CR(A0)    ; Reset MR Pointer
A   0000000C  117C00130001  MOVE.B #$13,MR1(A0)   ; No Parity, 8 bits/char.
A   00000012  117C00070001  MOVE.B #$07,MR2(A0)   ; 1 Stop Bit
A   00000018  117C00BB0003  MOVE.B #$BB,CSR(A0)   ;  9600-Baud XMIT and
                                                     RCV.
A   0000001E  117C00200005  MOVE.B #$20,CR(A0)    ; Reset the Receiver
A   00000024  117C00300005  MOVE.B #$30,CR(A0)    ; Reset the Transmitter
A   0000002A  117C00050005  MOVE.B #$05,CR(A0)    ; Enable XMIT and RCV.
```

Object: To Initialize Channel B

```
A   00000030  207C00F82010  MOVEA.L#$FE0010,A0   ; This is Channel B
                                                     Address
A   00000036  117C00100005  MOVE.B #$10,CR(A0)    ; Reset MR Pointer
A   0000003C  117C00130001  MOVE.B #$13,MR1(A0)   ; No Parity, 8 bits/char.
A   00000042  117C00070001  MOVE.B #$07,MR2(A0)   ; 1 Stop Bit
A   00000048  117C00BB0003  MOVE.B #$BB,CSR(A0)   ; 9600-Baud XMIT and
                                                     RCV.
A   0000004E  117C00200005  MOVE.B #$20,CR(A0)    ; Reset the Receiver
A   00000054  117C00300005  MOVE.B #$30,CR(A0)    ; Reset the Transmitter
A   0000005A  117C00050005  MOVE.B #$05,CR(A0)    ; Enable the XMIT and
                                                     RCV.
```

3.5 INTERRUPTS

The 68000 CPU on the XVME-601 Processor Module can be interrupted via all seven VMEbus interrupts, plus 2 local and 2 special interrupt sources. The local interrupt sources include the ABORT push-button, and the 68681 DUART. The special interrupts include ACFAIL* and SYSFAIL*. Note that ACFAIL* and SYSFAIL* are considered special interrupts in this case because they must be individually enabled by the CPU through the 68681 outputs. Refer to the 68681 Manual for information on the type of interrupts which the DUART itself can generate.

One of the outputs on the 68681 DUART is used to enable/disable all interrupt capability, and some are used to individually enable/clear local and special interrupt capabilities. Refer to Section 3.3 for information on controlling the 68681 outputs. Table 3-3 shows the interrupt sources and their corresponding VMEbus interrupt levels.

Table 3-3. XVME-601 Processor Module Interrupts

Interrupt Level	VMEbus Interrupts	Local Interrupts	Special Interrupts
Level 7	IRQ7*	ABORT Button	ACFAIL*
Level 6	IRQ6*		SYSFAIL*
Level 5	IRQ5*		
Level 4	IRQ4*	68681 DUART	
Level 3	IRQ3*		
Level 2	IRQ2*		
Level 1	IRQ1*		

When the 68681 DUART output OP7 is RESET, no 68000 interrupts can be generated. Output OP7 must be set in order for the CPU to receive any interrupts. The VMEbus interrupts IRQ1*-IRQ7* each have a corresponding jumper (refer to Section 2.4.4 of Chapter 2 for interrupt jumper information) which determines whether the corresponding interrupt source is enabled. The interrupt jumpers (J9-J15) are factory shipped in the "IN" position, with all interrupts enabled.

The SYSFAIL*, ABORT Button, and the ACFAIL* interrupts are all individually enabled/disabled via the CPU through certain 68681 DUART outputs (refer to Section 3-3). When these interrupts are detected, they are latched and they remain latched until the respective 68681 DUART output is RESET. The local/special interrupts have priority over the VMEbus interrupts when both occur on the same level. For example, an interrupt generated by pressing the ABORT Button would have priority over an IRQ7* interrupt coming from the bus. All local/special interrupts are auto vectored by interrupt level as defined by the Exception Vector Assignment of the MC68000 CPU. The state of all local interrupts can be polled via the input lines on the 68681 DUART. VMEbus interrupts are vectored on the Status I/O Byte returned by the interrupting VMEbus module, during the interrupt acknowledge (IACK) cycle.

The 68681 DUART interrupts can be enable/disabled via internal register bits (refer to the MC68681 Manual - Section 4 - Programming and Register Description).

Chapter 4

XVME-600MON
(OPTIONAL)

4.1 INTRODUCTION

The XVME-990/2 kit contains the 600MON debug monitor. It allows the user to design and debug firmware and software for the XVME-601 module.

The 600MON is a version of Motorola's VMEbug, modified to run on the XVME-601. It provides a powerful evaluation and debugging tool, permitting execution of system and user-developed programs under complete operator control. Debugging is eased by a complement of the features found on most debuggers plus an up/down load command for loading programs from a host, an assembler/disassembler for use in scanning and patching the code being debugged, and a self-test capability which verifies the integrity of the XVME-601.

This chapter discusses the 600MON in the following fashion:

- Referenced documents
- System configuration and I/O
- Instruction set
- User Routines
- Downloading

4.2 REFERENCED DOCUMENTS

It is assumed that the user has access to the following documents:

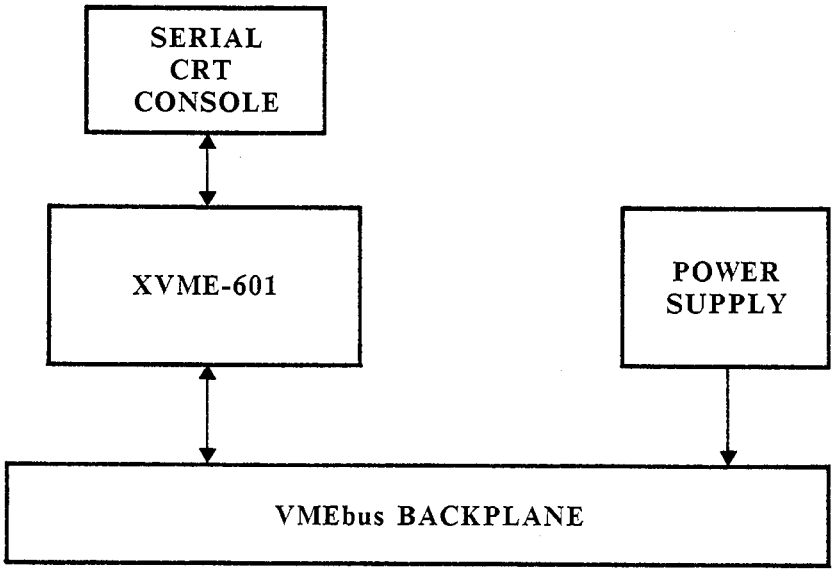
- Motorola VMEbug Debugging Package User's Manual MVMEBUG/D2
- M68000 Programmers Reference Manual - M68000UM(AD4)
- VMEbus Specification

4.3 SYSTEM CONFIGURATION

The minimum system configuration consists of:

- An XVME-601 Processor Module
- A VME backplane and power supply
- Serial Terminal
- Cable to interface the terminal to the XVME-601

The block diagram in Figure 4-1 illustrates the basic components required during the development phase.



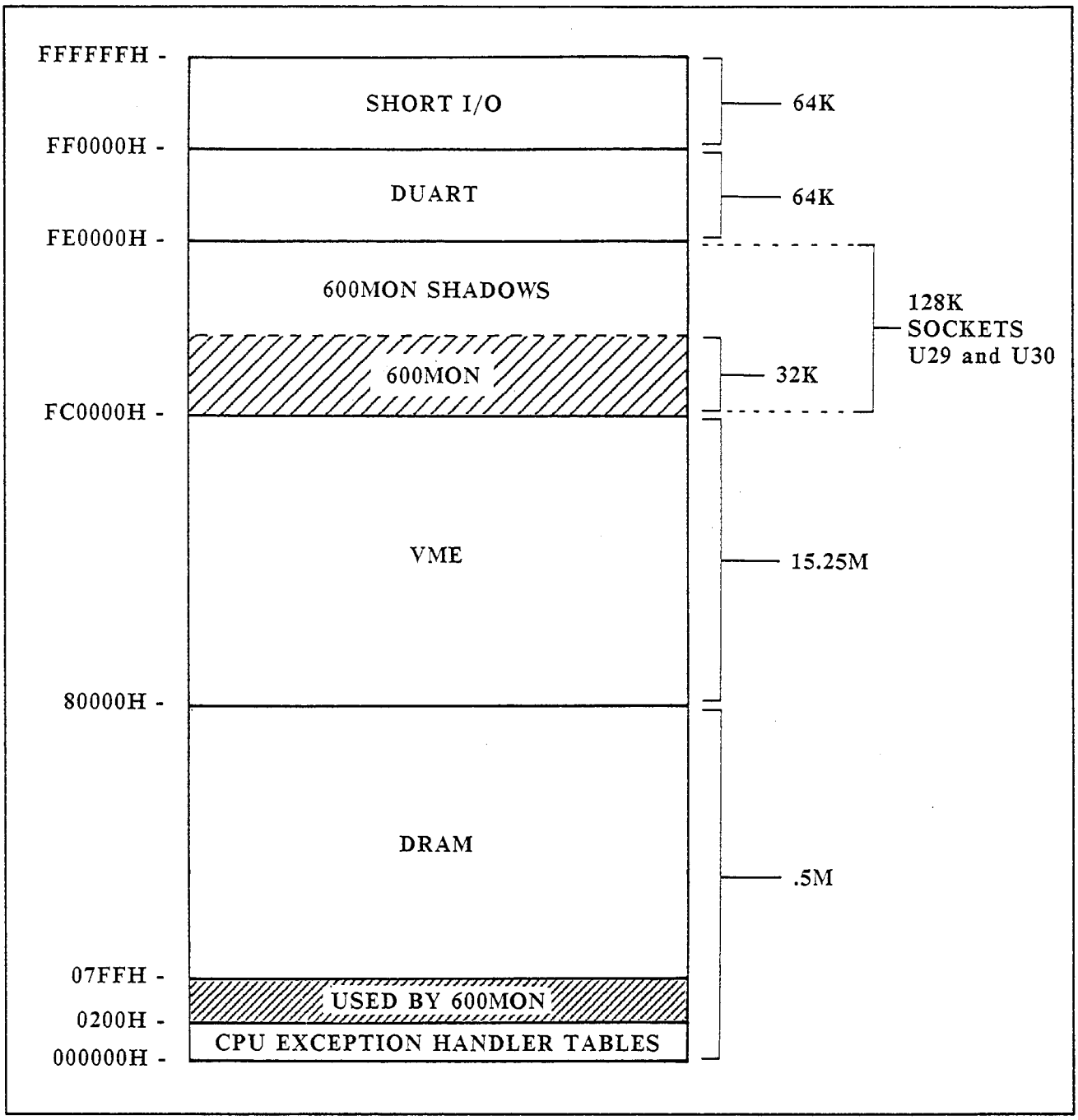
NOTES:

1. If the XVME-601 is used to provide the System Resource functions, it must occupy Slot 1 in the VMEbus, as prescribed by the VMEbus specifications.
2. If modules do not occupy consecutive slots in the backplane, the VMEbus backplane must be jumpered to continue both the IACKIN/IACKOUT and BGxIN/BGxOUT daisy chains.

Figure 4-1. Typical XVME-601 System Configuration

4.3.1 600MON Memory Map

The 600MON is burned into two 27128 EPROMs. The EPROMs have a 200nSec. access time. The 600MON uses the XVME-601's on-board DRAM from locations 000000H to 0007FFH. 600MON uses the XVME-601's memory as shown in Figure 4-2.



NOTES:

1. User programs should set the stack pointer(s) outside of the DRAM area used by 600MON.

2. 600MON will lock up when users try to access even addresses in the range FE0000H - FEFFFEH.

Figure 4-2. 600MON Memory Map

4.3.1.1 Exception Vectors

The 600MON exception handler table will reside in RAM locations 0000H through 01FFH.

This table is initialized by the monitor upon power-up and reset. The user can change the addresses of the handling routines. Exception vectors 0 through 127 are supported.

4.3.2 600MON I/O

600MON supports two serial channels numbered from one to two. Channel one is the local console and the default channel used in commands that can specify a channel for input and/or output. The RS-232 terminal must be connected to channel one.

The serial channels are initialized at power-up in the following format:

- 9600 baud
- 8-bits/character
- no parity

The configuration of a channel can be changed with the 600MON's PF command (see Section 4.6).

4.4 USING 600MON

Command entry is similar to that used in many buffer-organized systems. Typed characters are accumulated in the command line buffer by the input routine. A command is processed for execution following entry of a carriage return.

600MON provides limited editing capabilities for command entry. The delete (RUBOUT) key or CTRL H can be used to delete previously entered characters. CTRL X will cancel an entire line. CTRL D will re-display a command line.

During output to the console, CTRL W will suspend the output until another key is pressed. CTRL B will abort commands that perform console input/output (I/O). Variations in the basic function offered by many primitive commands can be obtained by entering appropriate characters in the options field of a command line. Some command functions are switched off by prefixing the command with NO. Commands and other I/O are presented in this manual using certain symbols. These symbols and their meanings are as follows:

<> Angular brackets enclose a symbol, known as a syntactic variable, that is replaced in a command line by one of a class of symbols it represents.

[] Square brackets enclose a symbol that is optional. The enclosed symbol may occur zero or one time.

[]... Square brackets followed by periods enclose a symbol that is optional/repetitive. The symbol may appear zero or more times.

In the examples given in the following paragraphs, operator entries are shown underscored for clarity only -- i.e., the underscore is not to be typed. Operator entries are followed by a carriage return unless otherwise specified.

The format of the command line is:

600MON x.x ><command>[<parameters>][;<options>]

where:

600MON x.x> Is the 600MON prompt (x.x represents the displayed version number).

command Is the primitive command.

parameter Is a field containing one of the following:

1. One or more expression parameters. Multiple expression parameters are separated by spaces.
2. One or more address parameters. Multiple address parameters are separated by spaces.
3. A data parameter opened with a space and contains one <pattern>, <data>, <count>, or <text> parameter.

option Is a field opened with a semicolon and containing one or more options. Multiple options need not be separated by spaces.

4.4.1 Expression Parameters

An expression can be one or more numeric values separated by the arithmetic operators plus (+) or minus (-). Numbers are assumed to be hexadecimal except for those preceded by an ampersand (&), which are decimal, and as noted for the DC command.

4.4.2 Address Parameters

An address, when used as a parameter, must follow the syntax accepted by the assembler except for the memory indirect mode. The address formats accepted are as follows:

FORMAT	EXAMPLE	DESCRIPTION
Address	140	Absolute address
Address + offset	130 + R5	Absolute address plus R5 offset value
(A@)	(A5)	Address register indirect with index
(A@,D@)	(A6,D4)	Address register indirect with index
Address (A@)	120 (A3)	Register indirect with displacement
Address (A@,D@)	110 (A2,D1)	Address register indirect with index plus displacement
[Address]	[100]	Memory indirect (not assembler-accepted syntax)

4.4.3 Offset Registers

The 600MON uses eight software registers, which are independent of the hardware, to modify addresses contained in 600MON commands.

The first seven registers (.R0-.R6) are used as general purpose offsets. The eighth register (.R7) is always zero.

These registers are modified by the .<register> command (see Section 4.6.1) and displayed by the Display Offsets command (OF) (see Section 4.6.19).

The offset registers are always reset to zero at power-up. Thus, if their contents are not changed, the registers have no effect on the entered address.

Unless another offset is entered, each command that expects an address parameter automatically adds offset R0 to the entered address. For example, if R0=100, the following commands are the same:

BR 10	(10 + 100)	R0 is added by default
BR 10 + R0	(10 + 100)	
BR 110 + R7	(110 + 0)	R7 is always zero

BR is the Breakpoint Set command (see Table B-2).
Note, the physical address for each of these commands is 110.

Offset R0 is automatically added to the offset registers any time they are modified. The only exception is, of course, when another offset register is specifically added. See Table 4-1 for examples.

Table 4-1. Offset Examples

.R0 0 + R7	(R0 = 0 + 0 = 0)	R0 set to zero
.R1 8	(R1 = 8 + 0 = 8)	Offset R0 is zero, R1 is set to 8
.R0 100	(R0 = 100 + 0 = 100)	Offset R0 added
.R0 200	(R0 = 200 + 100 = 300)	Offset R0 added
.R3 100 + R1	(R3 = 100 + 8 = 108)	Offset R0 not added
.R0 0 + R7	(R0 = 0 + 0 = 0)	R0 set to zero

4.4.4 Data Parameters

A parameter field containing an address parameter is always followed by a data field. Data field entries take one of the following forms:

DATA TYPE	USED WITH	DESCRIPTION
<pattern>	BF	A binary pattern expressed in up to four hex digits.
<data>	MS	1. An ASCII string of up to the length of the command line buffer (less (CR) character and preceding command line characters). 2. A value expressed in up to eight hex digits.
<count>	BR, MD, T	A value expressed in up to eight hex digits.
<text>	DU	An ASCII string of up to the length of the command line buffer (less (CR) character and preceding command line characters).

4.5 COMMAND VERIFICATION

As an aid to the user, 600MON displays for most commands its interpretation of the values entered as expression and address parameters. The results are displayed in a physical format. For example, if A000 had been typed in the parameter field of a command requiring that the contents of offset register R0 (1000, say) be added, then the resulting display would be 0000B000 for that address parameter.

Some checking of the validity of entered commands is also done. For example, specifying an address parameter which would result in an error causes the message "ADDRESS ERROR=00000000" to be displayed on the terminal connected to the serial port.

A table of 600MON error messages is provided in Appendix C.

4.6 COMMAND SET

This section describes the command line syntax and provides one or more examples for each command in the 600MON command set. Table 4-2 lists 600MON command mnemonics.

Table 4-2. 600MON Commands

COMMAND	DESCRIPTION
BF <address1><address2><pattern>	Block fill memory
BI <address1><address2>	Block initialize memory
BM <address1><address2><address3>	Block move memory
BR [<address>[;<count>]][<address>[;<count>]]...	Breakpoint set
NOBR [<address>[<address>]...]	Breakpoint remove
BS <address1><address2><data>[<mask>][;<option>]	Block search memory
BT <address1><address2>	Block test memory
DC <expression>	Data conversion
DF	Display registers formatted
DU [<port number>][<address1><address2>[<text>]	Dump memory as S-records
GD [<address>]	Go direct execute program
G[0][<address>]	Go execute program
GT <temp. breakpoint address>	Go until breakpoint
HE	Help
LO [<port number>][;[<options>]=<text>]	Load S-records into memory UB
MD [<port number>] <address> [<count][;DI]	Memory display/disassemble
M [M]<address>[;<options>]	Memory modify
MS <address><data>	Memory set
OF	Display offsets
PA [<port number>]	Port attach
NOPA [<port number>]	Port detach
PF [<port number>]	Port Format
R [M]	Register Modify
TM [<port number>][<exit character>[<trailing character>]]	Transparent mode
T [R] [<count>]	Trace
TT <breakpoint address>	Temp. breakpoint trace
VE [<port number>][;[<options>]=<text>]	Verify memory to S-records
.A0 - .A7 [<expression>]	Display/set address register
.D0 - .D7 [<expression>]	Display/set data register
.R0 - .R6 [<expression>]	Display/set relative offset
.PC [<expression>]	Display/set program counter
.SR [<expression>]	Display/set status register
.SS [<expression>]	Display/set s. stack pointer
.US [<expression>]	Display/set user stack pointer
(CTRL B)	Abort command
(DEL)	Delete character
(CTRL D)	Redisplay line

4.6.1 Display/Set Register

.<register> [<expression>]

Individual machine or offset registers can be displayed and/or altered by using the following format: .<register> [<expression>]. Commands with a leading period and the registers displayed/altered by these commands are:

.A0 - .A7	address register
.D0 - .D7	data register
.R0 - .R6	relative offset register (software register) (refer to OF command)
.PC	program counter
.SR	status register (in the MC68000)
.SS	supervisor stack pointer
.US	user stack pointer

See also: DF, OF

EXAMPLE	COMMENT
600MON x.x> .PC 5010	Set program counter.
600MON x.x> .A7 5300	Set address register seven.
600MON x.x> .R5 5500	Set relative offset register five.
600MON x.x> DF	"DF" is the Display Formatted Registers command (see Section 4.6.9)

```
PC=00005010   SR=FFFF=TS7XNZVC    US=00001000   SS=00005300
D0=0000D0D0  D1=0000D1D1  D2=0000D2D2  D3=00D3D3D3
D4=D4D4D4D4 D5=000000D5  D6=00000000  D7=000000D7
A0=000100A0  A1=001000A1  A2=0000A2A2  A3=000000A3
A4=000100A4  A5=001A5555  A6=000000A6  A7=00005300
-------------------005010          00000000       OR.B     #0,D0
```

4.6.2 Block Fill (BF)

BF <address1><address2><pattern>

The BF command fills a specified block of memory with a specified binary pattern of word size. A word boundary (even address) must be given for the starting <address1> and ending <address2> of the block. The pattern word may be expressed in hexadecimal (default), decimal, octal, or binary. The following symbols are used to denote number base:

$ precedes hexadecimal value (this is the default and may be omitted)
& precedes decimal value
@ precedes octal value
% precedes binary value

If a pattern of less than word size is entered, the data is right-justified and leading zeros are inserted by 600MON.

Example

```
600MON x.x> MD 4900
004900    FF FF 00 00 FF FF 00 00 FF FF 00 04 FF FF 00 00 ................

600MON x.x> BF 4900 490E 4E75
PHYSICAL ADDRESS = 00004900 0000490E

600MON x.x> MD 4900
004900    4E 75 4E 75 4E 75 4E 75 4E 75 4E 75 4E 75 4E 75 NuNuNuNuNuNuNuNu

600MON x.x>
```

4.6.3 Block Initialize (BI)

BI <address1><address2>

The BI command initializes word parity in a specified block of memory--<address1> through <address2>. No data in any word is changed if parity in the word is correct. If parity in a word is incorrect, the characters "m?" ($6D3F) are written in that word to force correct parity. If the parity cannot be set in one or more words, the message BUS TRAP ERROR is displayed on the console. The BT (Block Test) command may be used to isolate the failure(s).

NOTE

Both addresses must be on word boundaries.

See also: BT

Example

600MON x.x> BI 800000 80FFFE
PHYSICAL ADDRESS = 00800000 0080FFFE

600MON x.x.

4.6.4 Block Move (BM)

BM <address1><address2><address3>

The BM command is used to move (duplicate) blocks of memory from one area to another.

<address1> = beginning address of source memory block

<address2> = ending address of source memory block

<address3> = beginning address of destination memory block

Example

```
600MON x.x> MD 4900 A;DI
004900    1018          MOVE.B (A0)+,D0
004902    0C000000      CMP.B  #0,D0
004906    67F8          BEQ.S  $4A00
004908    4E75          RTS

600MON x.x> MD 4A00 A;DI
004A00    FFFF          DC.W      $FFFF
004A02    0000FFFF      OR.B      #-1,D0
004A06    0020FFFF      OR.B      #-1,-(A0)

600MON x.x> BM 4900 4909 4A00
PHYSICAL ADDRESS = 00004900 00004909
PHYSICAL ADDRESS = 00004A00

600MON x.x> MD 4A00 A;DI
004A00    1018          MOVE.B    (A0)+,D0
004A02    0C000000      CMP.B     #0,D0
004A06    67F8          BEQ.S     $4900
004A08    4E75          RTS

600MON x.x>
```

4.6.5 Breakpoint Set and Remove (BR and NOBR)

BR (display only)
BR [<address>[;<count>]][<address[;<count>]]...
NOBR [<address> [<address>]...]

When encountered, a breakpoint causes program execution to stop and control to be transferred to 600MON. The BR command may be used without parameters to cause display of current breakpoint addresses. The BR <address> command sets the address into the breakpoint address table. Multiple breakpoints (up to eight) may be specified with one call of the breakpoint command. Count is decremented each time the breakpoint is encountered until count=0; thereafter, execution will stop when the breakpoint is reached. the range of <count> is a 32-bit integer.

While executing a GO or GT command, the illegal instruction $4AFB is used for the breakpoint. If program control is lost and RESET is used to regain control, breakpoints may be left in the user target program.

While executing a Trace command, the breakpoint addresses are monitored (i.e., the illegal instruction 4AFB is not placed in memory).

After stopping at a breakpoint, execution may be continued by typing the GO command.

The NOBR command removes one or more breakpoints from the internal breakpoint table.

BR COMMAND FORMAT	DESCRIPTION
600MON x.x> BR	Display all breakpoints.
600MON x.x> BR <address>	Set a breakpoint.
600MON x.x> BR <address>;<count>	Set a breakpoint with a count.

See also: GT, TT

NOBR COMMAND FORMAT	DESCRIPTION
600MON x.x> NOBR	Clear all breakpoints.
600MON x.x> NOBR <address>	Clear a specific breakpoint.

See also: GT, TT

BR/NOBR Examples

```
600MON x.x> BR 4900 4A00 5300 5000

BREAKPOINTS
004900    004900
004A00    004A00
005300    005300
005000    005000

600MON x.x> NOBR 4A00 5000

BREAKPOINTS
004900    004900
005300    005300

600MON x.x> NOBR

BREAKPOINTS

600MON x.x> .R0 4000

BREAKPOINTS
000900+R0      004900
000A00+R0      004A00
000C00+F0      004C00

600MON x.x> NOBR A00

BREAKPOINTS
000900+R0      004900
000C00+R0      004C00

600MON x.x>
```

4.6.6 Block of Memory Search

BS <address1><address2> 'liter string'
BS <address1><address2><data>[<mask>][;<option>]

The BS command has two modes: 1) literal string search, and 2) data search. Both modes scan memory beginning at <address1> through <address2> looking for a match.

The literal string mode is initiated if a single quote (') follows <address2>. If a single quote does not follow <address2>, data search mode is assumed. In the data search mode, the optional mask, if used, is ANDed to data. The default mask is all ones. The options supported are:

;B byte
;W word
;L long word

The default is byte.

In both modes of the BS command, if the search finds matching data, the data and the address(es) are displayed. If the search is in data search mode with a mask, and data is found that matches the data after the mask is ANDed, the data from memory before applying the AND mask is displayed.

Example

```
600MON x.x> MD 1FF0 15
001FF0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF ..............
002000  43 43 45 45 00 00 00 00  00 00 00 00 00 00 00 00  CCEE........
```

```
600MON x.x> BS 1FF0 200F 'CC'
PHYSICAL ADDRESS = 00001FF0 0000200F
```

Successful search for literal string 'CC'.

```
600MON x.x> BS 1FF0 200F 34 ;W
PHYSICAL ADDRESS = 00001FF0 0000200F
```

Unsuccessful search for word length data (with default mask).

```
600MON x.x> BS 1FF0 200F 03 0F
PHYSICAL ADDRESS = 00001FF0 0000200F
002000    43
002001    43
```

Successful search for byte length data, with four most significant bits masked.

```
600MON x.x> BS 1000 3FFE 4AFB;W
PHYSICAL ADDRESS = 00001000 00003FFE
001000 4AFB
```

Successful search for "leftover" breakpoints.

```
600MON x.x>
```

4.6.7 Block of Memory Test (BT)

BT <address1><address2>

The BT command provides a destructive test of a block of memory. A word boundary (even address) must be given for the starting <address1> and ending <address2> of the block. If the test runs to completion without detecting an error, all memory tested will have been set to zeros.

Execution of this command may take several seconds for large blocks of memory.

When a problem is found in a memory location, the address, the data stored, and the data read are displayed, and control is returned to 600MON.

See also: BI

Example

```
600MON x.x> BT 1000 3FFE
PHYSICAL ADDRESS = 00001000 00003FFE

600MON x.x> BT 1000 3FFE
PHYSICAL ADDRESS = 001000 00003FFE
FAILED AT 003F00 WROTE=FFFF READ=0000

600MON x.x>
```

4.6.8 Data Conversion (DC)

DC <expression>

The DC command is used to convert an expression into hexadecimal and decimal. The expression may be entered in hexadecimal, decimal, or mixed format; output will be shown both ways. Default input format is hexadecimal. Octal and binary values may also be converted to decimal and hexadecimal values.

The following symbols are used:

$	precedes hexadecimal value (default; may be omitted)
&	precedes decimal value
@	precedes octal value
%	precedes binary value

Except for .R0, offset registers may not be used with the DC command.

This command is useful in calculating displacements such as destination of relative branch instructions or program counter relative addressing modes.

COMMAND FORMAT	DESCRIPTION
600MON x.x> DC $<data>	Convert hexadecimal data into hexadecimal and decimal.
600MON x.x> DC &<data>	Convert decimal data into hexadecimal and decimal

Example

```
600MON x.x> DC &120
$78=&120

600MON x.x> DC &15+$4-$13
$0=&0

600MON x.x> DC -1000
$FFFFF000=$1000=-&4096

600MON x.x> DC &15-$9+@14-%1100
$6=&6

600MON x.x>
```

4.6.9 Display Formatted Registers (DF)

DF

The DF command is used to display the CPU registers and the disassembled code of the instruction pointed to by the program counter. The registers display is also provided whenever the debugger gains control of the program execution -- i.e., at breakpoints and when tracing.

Note that any single register can be displayed with the .A0-.A7, .D0-.D7, etc., commands. See the display/set register command (.<register>), RM and the OF command.

Example

```
600MON x.x> DF
PC=000000 SR=2714=.S7..Z.. US=00001000 SS=00000C00
D0=00010434   D1=230A0444   D2=04060444   D3=00000000
D4=00010031   D5=0000072C   D6=00000004   D7=00000000
A0=00FE8001   A1=FFFFFFFF A2=00000454   A3=0000054E
A4=000131E0   A5=00002704   A6=00010158   A7=00000C00
----------000000          FABE                    DC.W  FABE

600MON x.x> .PC 001000

600MON x.x> .SS 003F00

600MON x.x> DF
PC=001000 SR=2714=.S7..Z.. US=00001000 SS=0003F00
D0=00010434   D1=230A0444   D2=04060444   D3=00000000
D4=00010031   D5=0000072C   D6=00000004   D7=00000000
A0=00FE8001   A1=FFFFFFFF A2=00000454   A3=0000054E
A4=000131E0   A5=00002704   A6=00010158   A7=00003F00
----------001000          000F      DC.W   $00F

600MON x.x>
```

4.6.10 Dump Memory (S-Records) (DU)

DU[<port number>]<address1><address2>[<text>]

The DU command formats memory data in S-record (serial formatted record) form and sends it to the specified port. The default port number is 1, the local display terminal. The first record output is an S0 record, which will contain the characters entered in the text field on the command line, if any. The last record output is an S9 record.

Valid port numbers for this command are:

none	defaults to 600MON port 1
n=1	specifies 600MON port 1
n=2	specifies 600MON port 2

This command does not send control characters to start or stop peripheral devices.

See also: LO, VE

Example

```
600MON x.x> DU 4000 400F 600MON TEST
PHYSICAL ADDRESS = 00004000 0000400F
S00E00003630304E4F4E205445535411354C7
S2140040000000000000000000000000000000000000AB
S9030000FC

600MON x.x>
```

4.6.11 Execute Program (G)

G[0][<address>]

The G (or GO) command causes the target program to execute (free run in real time) until:

a. The target program encounters a breakpoint,
b. An abnormal program sequence causes exception processing (e.g., divide by zero), or
c. The operator intervenes through use of the RESET or ABORT.

The G command has the following characteristics:

a. Starts execution at a give address or address pointed to by PC.
b. Traces first instruction.
c. Sets breakpoints set by BR.
d. Does not set temporary breakpoints.
e. The registers are set up as in the Display Format command.

NOTE

The execution period will exceed that of realtime time execution if breakpoints with count are encountered.

The G sequence starts by tracing one instruction, setting any breakpoints, and then free running.

COMMAND FORMAT	DESCRIPTION
600MON x.x> G	Begin execution at address in PC.
600MON x.x> GO <address>	Set PC = address and begin execution at that address.

See also: BR, DF, GD, GT, TR, TT

Example

```
600MON x.x> MD 4900 A;DI
004900          1018            MOVE.B   (A0)+,D0
004902          0C000000        CMP.B    #0,D0
004906          67F8            BEQ.S    $4900

600MON x.x> BR 4900 4908

BREAKPOINTS
004900    004900
004908    004908

600MON x.x> G 4900
PHYSICAL ADDRESS = 00004900

.

.

.
AT BREAKPOINT
PC=004908 SR=2700=.S7.... US=00001000 SS=00000C00
D0=00010402   D1=230A044    D2=04060444   D3=00000000
D4=00010031   D5=0000072C   D6=00000004   D7=00000000
A0=00FE8002   A1=FFFFFFFF   A2=00000454   A3=0000054E
A4=000131E0   A5=00002704   A6=00010158   A7=00000C00
------------004908          4E75          RTS

600MON x.x>
```

4.6.12 Go Direct Execute Program (GD)

GD[<address>]

The GD command is similar to the GO command, except that GD does not set breakpoints, nor does it start by tracing one instruction. The GD command has the following characteristics:

a. Starts execution at a given address or address pointed to by PC.
b. Does not trace first instructions.
c. Does not set breakpoints set by BR.
d. Does not set temporary breakpoints.
e. The registers are set up as in the Display Format command.

See also: GO, GT

Example

```
600MON x.x> MD 4900 A;DI
004900    1018              MOVE.B  (A0)+,D0
004902    0C000000          CMP.B   #0,D0
004906    66F8              BNE.S   $4900
004908    4E75              RTS

600MON x.x> BR 4900 4908

BREAKPOINTS
004900    004900
004908    004908

600MON x.x> G 4900
PHYSICAL ADDRESS = 00004900

AT BREAKPOINT
PC=004900 SR=2700=.S7.... US=000010000 SS=00000C00
D0=0001040C  D1=230A0444  D2=04060444  D3=00000000
D4=00010031  D5=0000072C  D6=00000004  D7=00000000
A0=00FE8003  A1=FFFFFFFF  A2=00000454  A3=0000054E
A4=000131E0  A5=00002704  A6=00010158  A7=00000C00
----------004908          1018       MOVE.B  (A0)+,D0

600MON x.x> GD 004900
PHYSICAL ADDRESS = 00004900

600MON x.x>
```

4.6.13 Go Until Breakpoint (GT)

GT <temporary breakpoint address>

The GT command has the following characteristics:

a. Starts execution at address pointed to by PC.
b. Does not trace first instruction.
c. Sets breakpoints set by BR.
d. Sets indicated temporary breakpoint.
e. The registers are set up as in the Display Format command.

When any breakpoint is encountered, the temporary breakpoint is reset.

If the address at which execution stops is in the breakpoint table, the message ERROR BRKPTS= is displayed.

See also: BR, DF, GD, GO, TR, TT

Example

```
600MON x.x> MD 4900 A;DI
004900    1018              MOVE.B   (A0)+,D0
004902    0C000000          CMP.B    #0,D0
004906    66F8              BNE.S    $4900
004908    4E75              RTS

600MON x.x> BR 4900 4908

BREAKPOINTS
004900    004900
004908    004908

600MON x.x> .PC 004900

600MON x.x> GT 004906
PHYSICAL ADDRESS = 00004906
PHYSICAL ADDRESS = 00004900

.
.
AT BREAKPOINT
PC=004906 SR=2700=.S7..Z.. US=00001000 SS=00000C04
D0=00010400   D1=230A0444   D2=04060444   D3=00000000
D4=00010031   D5=0000072C   D6=00000004   D7=00000000
A0=00FE8014   A1=FFFFFFFF   A2=00000454   A3=0000054E
A4=000131E0   A5=00002704   A6=00010158   A7=00000C04
----------004906          66F8       BNE.S     $4900

600MON x.x>
```

4.6.14 Help (HE)

HE

The HE command displays a list of available commands.

Example

```
600MON x.x> HE
.PC  .SR  .US  .SS
.D0  .D1  .D2  .D3  .D4  .D5  .D6  .D7
.A0  .A1  .A2  .A3  .A4  .A5  .A6  .A7
.R0  .R1  .R2  .R3  .R4  .R5  .R6  .R7

BF   BI   BM  BR   NOBR  BS  BT    DC
DF   DU   G   GD    GO   GT  HE    LO
M    MD   MM  MS    OF   PA  NOPA  PR
R    RM   T   TM    TR   TT  VE

600MON x.x>
```

4.6.15 Load (S-Records) (LO)

LO[<port number>][;[<options>]=<text>]

The LO command receives S-records from the specified port number and loads the contained object data into memory.

The following options are available:

;-C Ignore checksum while loading.

;X Echo data read from port 1, the local display terminal.

Text following the = is sent to the port

Valid port numbers for this command are:

none	defaults to 600MON port 1
n=1	specifies 600MON port 1
n=2	specifies 600MON port 2

The offset R0 is added to the load address specified in the S-records before storing the data. No control characters are sent by 600MON to control the transmission. Any line received not beginning with an 'S' is ignored. If an error occurs causing the system to take time to print out an error message, one or more lines sent during the error message may have been ignored.

A "." is printed on the console when each S-record is received.

See also: DU, OF, VE

Example

```
600MON x.x> LO2 ;=COPY FILE.SS,#CN02
COPY FILE.SS,#CN02 . . . . . .
```

4.6.16 Memory Display/disassemble (MD)

MD[<port number>]<address>[<count>][;DI]
MDS <address>[<count>][;DI]

The MD command is used to display a portion of memory beginning at <address> and displaying the number of bytes given as <count>. If <count> is not specified, the default is 10H bytes or, if MDS is used, the default number of bytes is 100H (a section). The display is in hex and in ASCII characters.

Default output is to port 1, the local display terminal. Specifying MD <port number> allows sending the output to another port.

Valid port numbers for this command are:

none	defaults to 600MON port 1
n=1	specifies 600MON port 1
n=2	specifies 600MON port 2

After the MD command is entered, it will continue with the next 16 lines of output each time a carriage return (CR) is entered. Any other command exits MD and enters the new command.

The ;DI option can be used to obtain source line disassembly of a specified or default number of bytes. If <count> is specified, disassembly of <count> bytes occurs. If MDS is used, disassembly of 32 bytes occurs. If <count> is not specified or MDS is not used, disassembly defaults to two or four bytes, depending on the address mode of the disassembled instruction. Note that invoking the ;DI option disables the continue-on-CR capability for an MD command line but not an MDS command line.

The ;DI option invokes the 600MON disassembler which transforms the bytes of machine code comprising the specified or default section of memory into lines of source code. Data is evaluated in several ways, depending on the context in which it is found. Addresses are displayed as hex numbers. Operands such as literals are displayed as decimal numbers. For example, the source line:

MOVE.L #1234,5678

assembles into the following eight hex digits:

21FC00001234172E

disassembly of which provides this line:

location 21FC00001234162E MOVE.L #4660,$0000162E

Note that the disassembled version differs from the original source line.

Many assemblers perform range checking for the purpose of optimizing the object code. Optimization occurs when an assembler, given a source instruction for which two forms exist, chooses the shorter form, thereby shortening the execution time. For example, if the value of the immediate data specified for an ADDI instruction is seven or less, the resident assembler will substitute an ADDQ instruction. In such cases, the disassembled version will differ from the original source instruction. Note that the 600MON one line assembler (see the Memory Modify command) does not perform range checking and, therefore, will not cause optimization differences between original and disassembled instructions.

In some cases, instructions having different mnemonic forms assemble into identical machine code. In such cases, the disassemble always chooses the same one of two mnemonics. For example, the resident assembler will produce the same code from the branch instructions BT (branch, condition true) and DBF (decrement and branch, condition false). The disassembler presents BRA when it encounters the former code and DBF when it encounters the latter code.

See also: MM, MA

Example

```
600MON x.x> MD 4900 3F
004900    20 3C 00 00 43 43 24 19   D6 82 0C 82 7F FF FF FF <..CC$.V.....
004910    67 02 60 08 06 83 00 00   00 01 60 E4 04 83 00 00   G.@.....@D....
004920    00 01 60 DC 00 00 00 00   00 00 00 00 00 00 00 00   ..@/........
004930    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ............

600MON x.x> MD 4900 12;DI
004900    203C00004343          MOVE.L   #17219,D0
004906    2419                  MOVE.L   (A1)+,D2
004908    D682                  ADD.L    D2,D3
00490A    0C827FFFFFFF          CMP.L    #2147483647,D2
004910    6702                  BEQ.S    $4914

600MON x.x>

600MON x.x> MD2 5400 2;DI
(output is sent to port 2)

600MON x.x>

600MON x.x> MDS 5400 2;DI
005400    00000000        OR.B      #0,D0

600MON x.x>
```

4.6.17 Memory Modify/Disassemble/Assemble (M)

M[M]<address>[;<options>]

The function of the M or MM command is to change data in memory. An address and options are specified on the initial command line. On execution, one of two major modes of operation is entered -- the memory change mode or the disassemble/assemble mode.

Memory Change Mode

For convenient viewing and changing of object data, this mode offers four variations of data updating capability. These are enhanced by five options: the data size options, word and long word (the default size is byte); odd or even address access options (byte size only); and a non-verification option for write-only operations. Action provided by an option specified on the initial command line is utilized in all four data updating submodes and remains in force until the M command is exited.

The five memory change mode options are:

;W	Set size to word (i.e., two bytes).
;L	Set size to long word (i.e., four bytes).
;O	Set size to byte; access only odd addresses.
;V	Set size to byte; access only even addresses.
;N	No verification. Do not read data after updating. If used, ;N must be preceded by one of above options (the semicolon (;) is required between multiple options).

When the memory change mode is entered on execution of the initial command line, object data in the specified locations is displayed in hexadecimal format and the M command prompt (?) is presented at the right of the data. The data can then be changed, using any of the subcommands described below. If desired, the action of the subcommand can be obtained without entering new data. For example, the contents of the preceding locations(s) can be viewed by typing "^(CR)" alone after the ? prompt, or the M command can be exited by typing ".(CR)".

[<data>](CR)	Update location and sequence forward.
[<data>]^(CR)	Update location and sequence backward.
[<data>]=(CR)	Update location and reopen same location.
[<data>].(CR)	Update location and terminate.

Disassemble/Assemble Mode (the ;DI option)

On execution of an initial M command line with the ;DI option selected, the disassemble/assemble mode is entered. Starting from the specified location, data is disassembled into a source instruction line, and both object data (in hexadecimal) and the source line are displayed. The M command prompt (?) is displayed to the right of the disassembled source line. If desired, a new source instruction may be typed and assembled to replace the existing instruction. (The first character must be a space, which is recognized as the label field delimiter by the 600MON on line assembler.) Assembly is initiated by typing a carriage return. After assembly and updating, data in the following locations is disassembled and the next source line displayed. Note that the update and sequence backward, and the update and reopen the same location features, are not available in the disassemble/assemble mode. Typing ".(CR)" while in this mode will also provide exit from the M command.

See also: MD, MS

Example

```
600MON x.x> M 5000 ;L
00005000 00000000 ? 200=
00005000 00000200 ? .

600MON x.x> MM 5400;W;N
00005400 ? 555
00005402 ? 34.

600MON x.x> M 5000;DI
005000    48B800010406        MOVEM.WD0,$0406 ?
005006    40F80406            MOVE.W SR,$0406 ?
00500A    48E7FFFE            MOVEM.LD0-D7/A0-A6,-(A7) ?
00500E    4FF8095A            LEA.L    $095A,A7 ?MOVE.B

00500E    4FF8095A    X?                          (response to incomplete,
00500E    4FF8095A    LEA.L    $095A,A7 ?         incorrect entry)
005012    1600        MOVE.B   D0,D3 ?
005014    04444281    SUB.W    #17025,D4 ?.
```

4.6.18 Memory Set (MS)

MS <address><data>

The function of the MS command is to change data in memory. Data to be written and the starting location are specified on the MS command line. Existing data is overwritten.

Data is specified as bytes, words, and long words. Hexadecimal data of different lengths can be intermingled on the MS command line. One or more ASCII characters also may be specified as data. ASCII characters are enclosed by quotation marks.

Data up to the capacity of the command buffer can be written into sequential memory locations with a single invocation of MS. Buffer capacity is 128 characters. The command automatically sends a CR/LF sequence to the console, allowing additional characters to be specified on the next line.

See also: MD, MM

Example

600MON x.x> MS 2000 'ABC' — Set memory to ASCII string.

600MON x.x> MS 2000 4445 — Set memory to hexadecimal word data.

600MON x.x> MS 2005 12345678 12 — Set memory to hexadecimal long word, byte data.

```
600MON x.x> MD 1000 10
001000    00 00 3f ff 00 00 00 00 00 00 00 00 00 00 00 00      ../.....

600MON x.x> MS 1000 'ABCDEFGH'

600MON x.x> MD 1000 10
001000    41 42 43 44 45 46 47 48 00 00 00 00 00 00 00 00      ABCDEFGH.....

600MON x.x>
```

4.6.19 Display Offsets (OF)

OF

The OF command displays the offsets used to assist with relocatability and position-independent code.

Linked segments of code will each have different load address or offset. For user convenience, seven general purpose offsets (.R0-.R6) are provided. Offset .R7 is always zero, which provides a convenient technique of entering an address without an offset. If no value is assigned to one of the general purpose offsets, it will have the default value of zero.

Unless another offset is entered, each command that expects an address parameter automatically adds offset R0 to the entered address -- that is, if R0=1000, the following commands are the same:

```
BR 10
BR 10+R0
BR 1010+R7
```

Example

```
600MON x.x> .R1 1000                          Set offset R1.

600MON x.x> .R3 3300

600MON x.x> .R4 440000

600MON x.x> .R5 0                             Set offset R5.

600MON x.x> OF                                Display offsets.
R0=00000000 R1=00001000 R2=00000000 R3=00003300
R4=00440000 R5=00000000 R6=FFFFFFFF R7=00000000

600MON x.x> .R0 2000                          Set offset R0.

600MON x.x> MM10
000010+R0 61 ?                                Offset R0 is added to the address.

600MON x.x> MM 10+R7                          R7 is always 0, so it "overrides"
000010 00                                     R0.

600MON x.x>
```

NOTE

To set R0 to 0 after is has been set to a nonzero value, use the command ".R0 0+.R7". The command ".R0 0" will not alter R0.

4.6.20 Port Attach and Detach (PA and NOPA)

PA[<port number>]
NOPA

The PA command allows the user to attach a poet so that information sent to the port 1 terminal will be echoed on another port.

Valid port numbers for this command are:

none	defaults to 600MON port 1
n=1	specifies 600MON port 1
n=2	specifies 600MON port 2

A serial printer could be connected to one of the ports so data printed on the port 1 terminal will also be printed on the printer attached to one of the other ports.

Hard copy of a debug session can be used to communicate complex problems to other users.

NOTE

Only one port can be attached at a time.

4.6.21 Port Format (PF)

PF[<port number>]

Use of the command PF with no specified port number n causes display of the current configuration of each port. To change the current configuration of a specific port, the port number is specified with the PF<port number> command.

Valid port number for this command are:

n=1	allows changing configuration of 600MON port 1
n=2	allows changing configuration of 600MON port 2

Examples

```
600MON x.x> PF2
BAUD RATE  =0B ? 09
BITS/CHAR  =08 ? 7
PARITY     =04 ? 0
CHAR NULL  =00 ? 1
CR NULL    =00 ? 3

600MON x.x> PF2
BAUD RATE  =09 ?
BITS/CHAR  =07 ?
PARITY     =04 ?
CHAR NULL  =01 ?
CR NULL    =03 ?
```

The BAUD RATE, BITS/CHAR, and PARITY parameters are used to reconfigure serial ports on the XVME-601.

The CHAR NULL and CR NULL parameters specify the number of nulls to be sent after each character and each line.

The following tables show the codes to use for the various port configuration options:

BAUD RATES		BITS/CHAR	PARITY
0=50	6=1200	5=5	0=Even
1=110	7=1050	6=6	1=Odd
2=134.5	8=2400	7=7	2=0
3=200	9=4800	8=8	3=1
4=300	A=7200		4=none
5=600	B=9600		
	C=38.4K		

4.6.22 Register Modify

R[M]

The R command allows the user to view and modify the data and address registers individually. Each time the carriage return is pressed, the next register is displayed and the user is given the option to change it. The registers are displayed in the order D0-D7 and A0-A7.

Example

```
600MON x.x> R
D0=0000011C ?
 1=00000010 ?
 2=00004C4F ?
 3=00000000 ?
 4=00000000 ?  00100800
 5=00000000 ?
 6=00000003 ?  00100000
 7=00000000 ?
A0=008020F2 ?
 1=00000D4C ? 00100A00
 2=00000800 ?
 3=0010053F ?  00100B00
 4=00100545 ?
 5=0010053E ?  00100C00
 6=0010054E ?
 7=0000079C ?

600MON x.x> R
D0=0000011C ?
 1=00000010 ?
 2=00004C4F ?
 3=00000000 ?
 4=00100800 ?
 5=00000000 ?
 6=00100000 ?
 7=00000000 ?
A0=008020F2 ?
 1=00100A00 ?
 2=00000800 ?
 3=00100B00 ?
 4=00100545 ?
 5=00100C00 ?
 6=0010054E ?
 7=0000079C ?

600MON x.x>
```

4.6.23 Transparent Mode (TM)

TM[<port number>][<exit character>[<trailing character>]]

The TME command connects port 2 to port 1 and transmits all input/output between them until the exit character is entered from the terminal attached to port 1. The default exit character is CTRL A($01).

A trailing character is sent to the host after the exit character is encountered in order to remove the exit character from the host's buffer so that it will not be encountered the next time transparent mode is entered. The default trailing character is CTRL X ($18). Default exit and trailing characters can be changed by entering the following along with the TM command: TM x y, where x is the new exit character and y is the new trailing character. If y=null, then it is not sent to the host. To enter a CTRL-character combination, the CTRL key and the character key are depressed simultaneously.

For optimum operation, the device connected to port 2 should operate at the same baud rate as the device connected to port 1. The port 2 device can operate more slowly than the port 1 terminal but if the port 1 terminal is set slower than the port 2 device, the system probably will not work.

Valid port numbers for this command are:

n=2 specifies 600MON port 2

See also: LO, VE

Example	Comments
600MON x.x>	Start-up or reset condition.
600MON x.x> TM	Command to enter transparent mode.
TRANSPARENT EXIT=$01 = CTRL A	600MON prints this: Exit=$01 means that in order to exit this mode, the operator must enter CTRL A.

User talks directly to host, uses editor, assembler, etc.

(CTRL) A	Ends the transparent mode.
600MON x.x>	600MON prints its prompt and system is ready for new command.

4.6.24 Trace (T)

T[R] [<count>]

The T (or TR) command executes instructions, one at a time, beginning at the location pointed to by the program counter. After execution of each instruction, the MPU registers are displayed.

Once the trace mode is entered, the prompt includes a colon (i.e., 600MON x.x>). While in this mode, typing the single character (CR) will cause one instruction to be traced.

Breakpoints and breakpoint counts are in effect during trace.

Trace cannot be used to step through interrupts or exceptions (TRAP, etc.).

COMMAND FORMAT	DESCRIPTION
600MON x.x> T	Trace one instruction.
600MON x.x> TR <count>	Trace "count" instructions.
600MON x.x> (CR)	Carriage return (CR) executes next instruction.
600MON x.x> .	Typing a period (.) followed by a (CR) exits trace mode.

NOTE

If the program counter contains an address that falls between the starting and ending addresses of the 600MON program, a warning message, .PC within "debugger", will be returned. Processing will continue but with unexpected if stack pointers and/or registers are not handled properly.

See also: DF, GO, GT, TT

Examples

```
600MON x.x> .PC 5000

600MON x.x> TR
PHYSICAL ADDRESS=00005000
PC=005002 SR=2709=.S7.N..C    US=00001000   SS=00000C00
D0=00300034   D1=00004D4D  D2=00010004   D3=00000000
D4=00000F37   D5=00000000  D6=00000001   D7=00000000
A0=000100CA   A1=00010510  A2=00000546   A3=00000000
A4=00002004   A5=0000053A  A6=00000544   A7=00000C00
----------005002                4E71         NOP

600MON x.x> T2
PHYSICAL ADDRESS=00005002
PC=005004 SR=2709=.S7.N..C    US=00001000   SS=00000C00
D0=00300034   D1=00004D4D  D2=00010004   D3=00000000
D4=00000F37   D5=00000000  D6=00000001   D7=00000000
A0=000100CA   A1=00010510  A2=00000546   A3=00000000
A4=00002004   A5=0000053A  A6=00000544   A7=00000C00
----------005004                4E71         NOP
PC=005006 SR=2709=.S7.N..C    US=00001000   SS=00000C00
D0=00300034   D1=00004D4D  D2=00010004   D3=00000000
D4=00000F37   D5=00000000  D6=00000001   D7=00000000
A0=000100CA   A1=00010510  A2=00000546   A3=00000000
A4=00002004   A5=0000053A  A6=00000544   A7=00000C00
----------005006                4E71         NOP

600MON x.x> .

600MON x.x>
```

4.6.25 Trace to Temporary Breakpoint (TT)

TT <breakpoint address>

The TT command performs the following:

1. Sets a temporary breakpoint at the address specified.
2. Starts a program execution in the trace mode.
3. Traces until any breakpoint with a zero count is encountered.
4. Resets the temporary breakpoint.

The temporary breakpoint is not displayed by the BR command.

See also: DF, GO, GT, TR

Example

```
600MON x.x> .PC 5000

600MON x.x> TT 5006
PHYSICAL ADDRESS=00005006
PHYSICAL ADDRESS=00005000
PC=005002 SR=2709=.D7.N..C   US=00001000   SS=00000C00
D0=00300034   D1=00004D4D   D2=00010004   D3=00000000
D4=00000F37   D5=00000000   D6=00000001   D7=00000000
A0=000100CA   A1=00010510   A2=00000546   A3=00000000
A4=00002004   A5=0000053A   A6=00000544   A7=00000C00
----------005002                4E71          NOP
PC=005004 SR2709.S7.N..C     US=00001000   SS=00000C00
D0=00300034   D1=00004D4D   D2=00010004   D3=00000000
D4=00000F37   D5=00000000   D6=00000001   D7=00000000
A0=000100CA   A1=00010510   A2=00000546   A3=00000000
A4=00002004   A5=0000053A   A6=00000544   A7=00000C00
----------005004                4E71          NOP

AT BREAKPOINT
PC=005006 SR=2709=.S7.N..C   US=00001000   SS=00000C00
D0=00300034   D1=00004D4D   D2=00010004   D3=00000000
D4=00000F37   D5=00000000   D6=00000001   D7=00000000
A0=000100CA   A1=00010510   A2=00000546   A3=00000000
A4=00002004   A5=0000053A   A6=00000544   A7=00000C00
----------005006                4E71          NOP

600MON x.x> .

600MON x.x>
```

4.6.26 Verify (S-Records) (VE)

VE[<port number>][;=<text>]

The VE command verifies the current contents of memory with the object data in S record format received from a serial port. When a mismatch between data in memory and the S-record object data is found, 600MON will display the differences on the port 1 terminal. If display of differences occurs, the record following the record displayed is lost. Text following the = is sent to port 2.

Transmitted object data can be verified through the same port through which it was sent.

Valid port numbers for this command are:

none	defaults to verify data from 600MON port 1
n=1	specifies verifying data from 600MON port 1
n=2	specifies verifying data from 600MON port 2

See also: DU, LO, OF

Example	Comment
600MON x.x> VE ;=COPY TEST1.MX,# 21231000-.-.49-.4E-.-.-.-.-.-.-.-.-.-.-.-. 600MON x.x>	Verify the file TEST1.MX. The record is an S1 record with 23 byte-count starting at address 0010000, and there are differences in the third and fifth bytes. (The difference in data displayed is the data being read, not the data from memory.)

4.7 USING 600MON I/O ROUTINES THROUGH TRAP 15

The user calls an I/O routine by having the program perform the following sequence:

```
TRAP #15      Call to 600MON trap handler
DC.W $00xx    Function being requested (x = function)
```

Table 4-3 gives the available I/O routines and Table 4-4 give the registers used by the function.

Table 4-3. Available User Routines

FUNCTION	DESCRIPTION
0	Return to monitor.
1	Input Line - Channel 1.
2	Output Line +CR/LF- Channel 1.
3	Input Line - Channel 2.
4	Output Line +CR/LF- Channel 2.
5	Not Used.
6	Output Line - Channel 1.
7	Output Line - Channel 2.
8	Not Used.
9	Not Used.
A	Not Used.
B	Not Used.
C	Not Used.
D	Not Used.
E	Not Used.
F	Not Used.
10	Input Status
11	Input Character Without Echo
12	Input Character With Echo
13	Input Line Without Echo
14	Input Line With Echo
15	Output Character
16	Output Line
17	Output Line +CR/LF

Table 4-4. Registers Used by Function

FUNCTION(s)	REGISTERS USED	
1,3	Entry:	Beginning of line in A5.L, A6.1
	Exit:	Beginning of line in A5.L, end of line +1 in A6.L
2,4,6,7	Entry:	Beginning of line in A5.L, end of line +1 in A6.L
10	Entry:	Channel # in D0.B (1 or 2)
	Exit:	Status in D1.B (00H=no char. ready, FFH=char. ready)
11,12	Entry:	Channel # in D0.B (1 or 2)
	Exit:	Character in D1.B
13,14	Entry:	Channel # in D0.B (1 or2) Beginning of line in A5.L, A6.L
	Exit:	Beginning of line in A5.L, end of line +1 in A6.L
15	Entry:	Channel # in D0.B (1 or 2) Character in D1.B
16,17	Entry:	Channel # in D0.B (1 or 2) Beginning of line in A5.L, end of line +1 in A6.L

Appendix A

VMEbus CONNECTOR/PIN DESCRIPTION

The XVME-601 Processor Module is a single-high VMEbus compatible board. There is one 96-pin bus connector on the rear edge of the board labeled P1 (refer to Chapter 2, Figure 2-1 for the location). The signals carried by connector P1 are the standard address, data, and control signals required for a P1 backplane interface, as defined by the VMEbus specification. Table A-1 identifies and defines the signals carried by the P1 connector.

Table A-1. P1 - VMEbus Signal Identification

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
ACFAIL*	1B:3	AC FAILURE: Open-collectors driven signal which indicates that the AC input to the power supply is no longer being provided, or that the required input voltage levels are not being met.
IACKIN*	1A:21	INTERRUPT ACKNOWLEDGE IN: Totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chained acknowledge. The IACKIN* signal indicates to the VME board that an acknowledge cycle is in progress.
IACKOUT*	1A:22	INTERRUPT ACKNOWLEDGE OUT: Totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chained acknowledge. The IACKOUT* signal indicates to the next board that an acknowledge cycle is in progress.
AM0-AM5	1A:23 1B:16,17, 18,19 1C:14	ADDRESS MODIFIER (bits 0-5): Three-state driven lines that provide additional information about the address bus, such as: size, cycle type, and/or DTB master identification.
AS*	1A:18	ADDRESS STROBE: Three-state driven signal that indicates a valid address is on the address bus.

Table A-1. VMEbus Signal Identification (cont'd)

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
A01-A23	1A:24-30 1C:15-30	ADDRESS BUS (bits 1-23): Three-state driven address lines that specify a memory address.
A24-A31	2B:4-11	ADDRESS BUS (bits 24-31): Three-state driven bus expansion address lines.
BBSY*	1B:1	BUS BUSY: Open-collector driven signal generated by the current DTB master to indicate that it is using the bus.
BCLR*	1B:2	BUS CLEAR: Totem-pole driven signal generated by the bus arbitrator to request release by the DTB master if a higher level is requesting the bus.
BERR*	1C:11	BUS ERROR: Open-collector driven signal generated by a slave. It indicates that an unrecoverable error has occurred and the bus cycle must be aborted.
BG0IN*- BG3IN*	1B:4,6, 8,10	BUS GRANT (0-3) IN: Totem-pole driven signals generated by the Arbiter or Requesters. Bus Grant In and Out signals form a daisy-chained bus grant. The Bus Grant In signal indicates to this board that it may become the next bus master.
BG0OUT*- BG3OUT*	1B:5,7, 9,11	BUS GRANT (0-3) OUT: Totem-pole driven signals generated by Requesters. These signals indicate that a DTB master in the daisy-chain requires access to the bus.

Table A-1. VMEbus Signal Identification (cont'd)

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
BR0*-BR3*	1B:12-15	BUS REQUEST (0-3): Open-collector driven signals generated by Requesters. These signals indicate that a DTB master in the daisy-chain requires access to the bus.
DS0*	1A:13	DATA STROBE 0: Three-state driven signal that indicates during byte and word transfers that a data transfer will occur on data buss lines (D00-D07).
DS1*	1A:12	DATA STROBE 1: Three-state driven signal that indicates during byte and word transfers that a data transfer will occur on data bus lines (D0-D15).
DTACK*	1A:16	DATA TRANSFER ACKNOWLEDGE: Open-collector driven signal generated by a DTB slave. The falling edge of this signal indicates that valid data is available on the data bus during a read cycle, or that data has been accepted from the data bus during a write cycle.
D00-D15	1A:1-8 1C:1-8	DATA BUS (bits 0-15): Three-state driven, bi-directional data lines that provide a data path between the DTB master and slave.
GND	1A:9,11, 15,17,19, 1B:20,23, 1C:9 2B:2,12, 22,31	GROUND

Table A-1. VMEbus Signal Identification (cont'd)

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
IACK*	1A:20	INTERRUPT ACKNOWLEDGE: Open-collector or three-state driven signal from any master processing an interrupt request. It is routed via the backplane to slot 1, where it is looped-back to become slot 1 IACKIN* in order to start the interrupt acknowledge daisy-chain.
IRQ1*-IRQ7*	1B:24-30	INTERRUPT REQUEST (1-7): Open-collector driven signals, generated by an interrupter, which carry prioritized interrupt requests. Level seven is the highest priority.
LWORD*	1C:13	LONGWORD: Three-state driven signal indicates that the current transfer is a 32-bit transfer.
(RESERV-ED)	2B:3	RESERVED: Signal line reserved for future VMEbus enhancements. This line must not be used.
SERCLK	1B:21	A reserved signal which will be used as the clock for a serial communication bus protocol which is still being finalized.
SERDAT	1B:22	A reserved signal which will be used as the transmission line for serial communication bus messages.
SYSCLK	1A:10	SYSTEM CLOCK: A constant 16-MHz clock signal that is independent of processor speed or timing. It is used for general system timing use.

Table A-1. VMEbus Signal Identification (cont'd)

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
SYSFAIL*	1C:10	SYSTEM FAIL: Open-collector driven signal that indicates that a failure has occurred in the system. It may be generated by any module on the VMEbus.
SYSRESET*	1C:12	SYSTEM RESET: Open-collector driven signal which, when low, will cause the system to be reset.
WRITE*	1A:14	WRITE: Three-state driven signal that specifies the data transfer cycle in progress to be either read or written. A high level indicates a read operation, a low level indicates a write operation.
+5V STDBY	1B:31	+5 VDC STANDBY: This line supplies +5 VDC to devices requiring battery backup.
+5V	1A:32 1B:32 1C:32 2B:1,13,32	+5 VDC POWER: Used by system logic circuits.
+12V	1C:31	+12 VDC POWER: Used by system logic circuits.
-12V	1A:31	-12 VDC POWER: Used by system logic circuits.

BACKPLANE CONNECTOR P1

The following table lists the P1 pin assignments by pin number order. (The connector consists of three rows of pins labeled rows A, B, and C.)

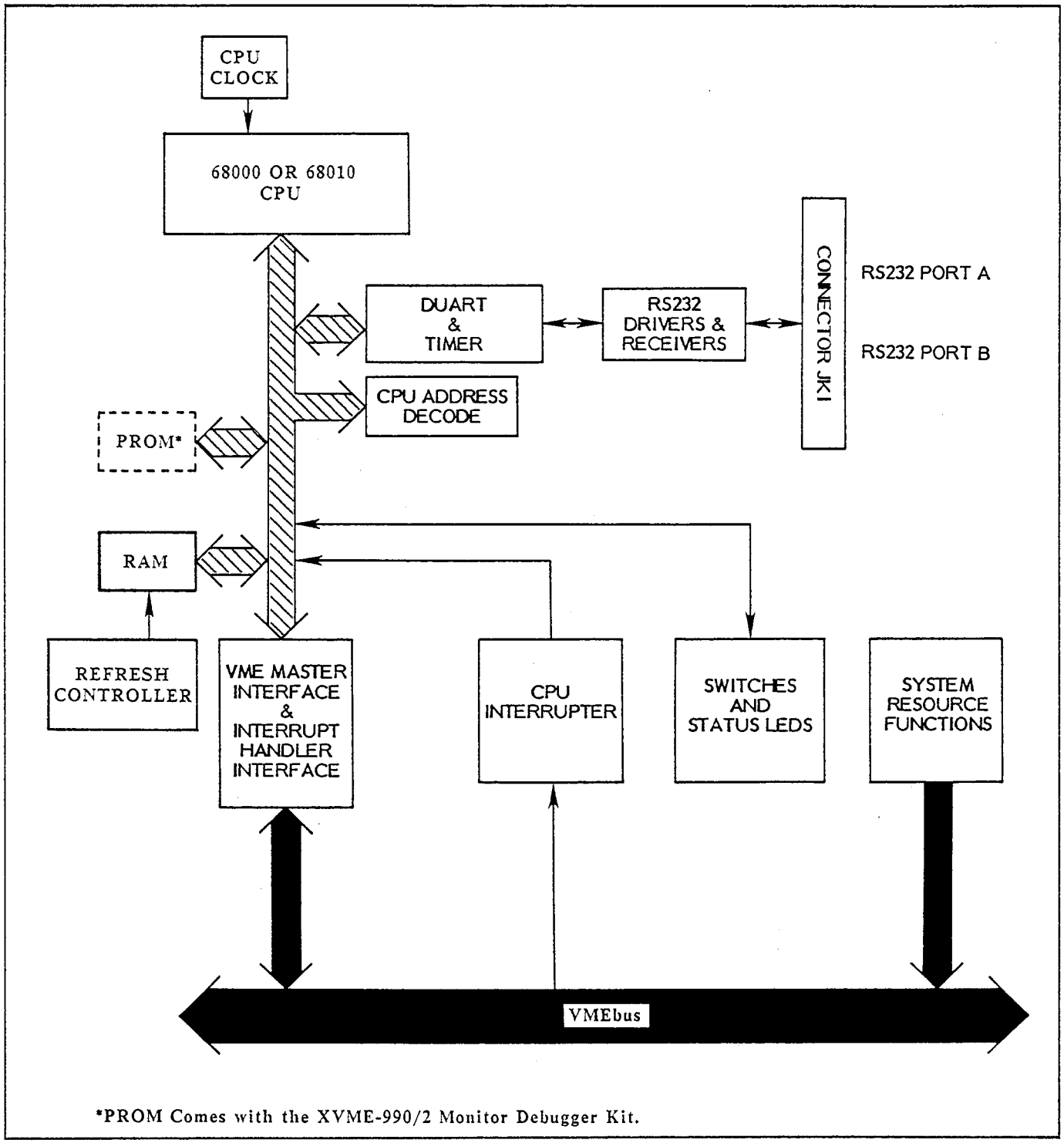
Table A-2. P1 Pin Assignments

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK(1)	A17
22	IACKOUT*	SERDAT(1)	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5V STDBY	+12V
32	+5V	+5V	+5V

Appendix B

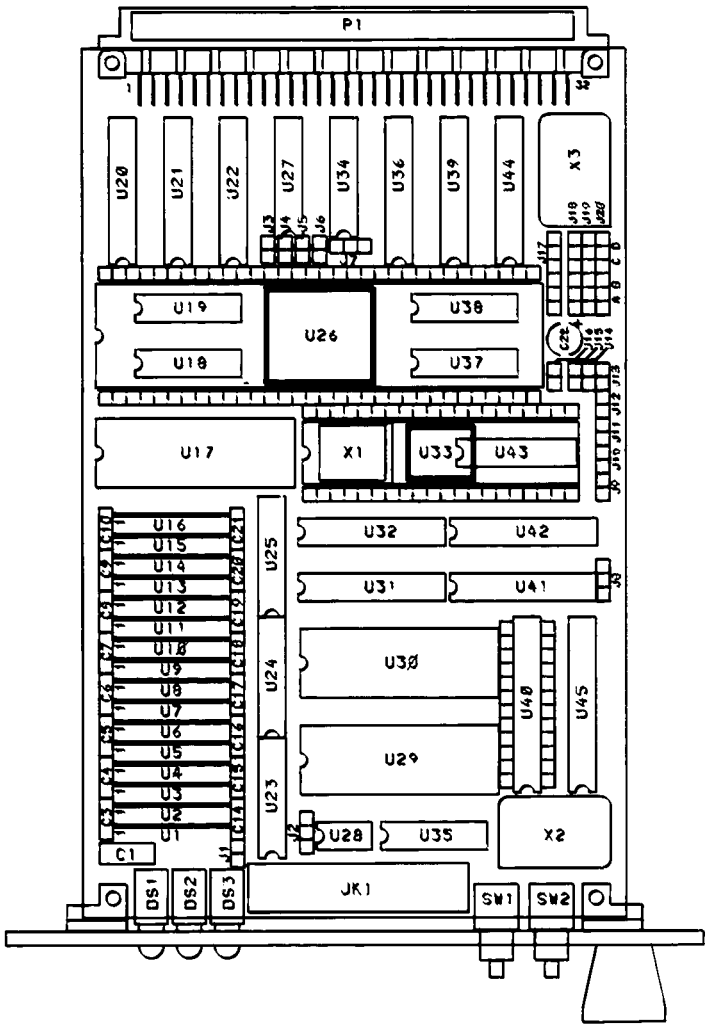
BLOCK DIAGRAM, ASSEMBLY DRAWING, & SCHEMATICS

BLOCK DIAGRAM



*PROM Comes with the XVME-990/2 Monitor Debugger Kit.

ASSEMBLY DRAWING



REMOVE THIS SHEET!

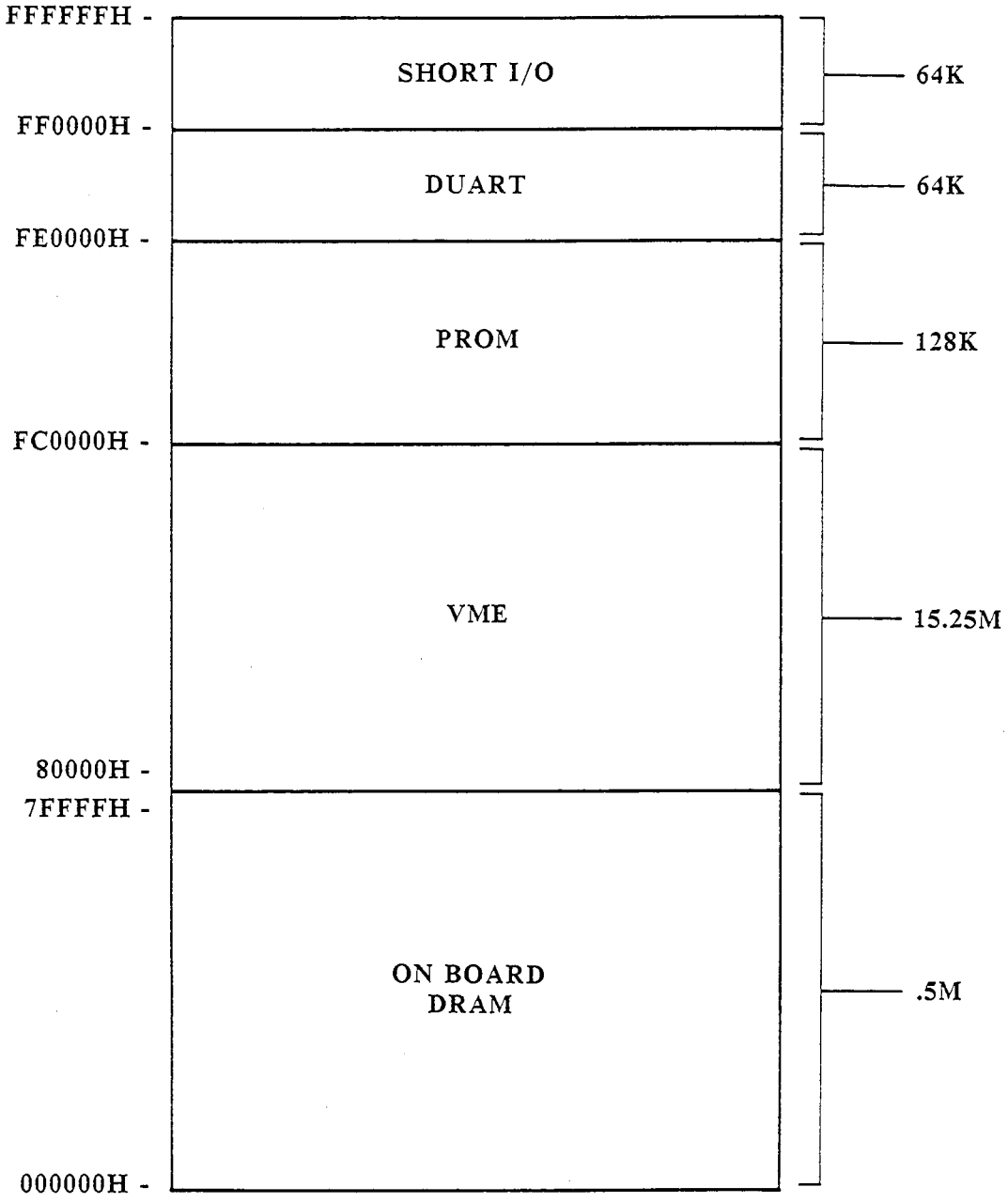
Insert Schematic Sheet Here

REMOVE THIS SHEET!

Appendix C

QUICK REFERENCE GUIDE

Memory Map



Jumper List

Jumper	Use
J1	This jumper is used to enable/disable the reset switch on the front panel of the module.
J3,J4,J5,J6,J7, and J17	These jumpers are used to enable/disable the system resource functions
J2,J8	These jumpers are configured to match the type and size of EPROM which is installed on the XVME-601.
J9,10,11,12,13,14,15	These jumpers are used to determine which VMEbus interrupt levels (1-7) can be used to interrupt the CPU.
J16	This jumper enables/disables the abort switch on the front panel of the module.
J18,J19, & J20	These jumpers are used to select the bus request and bus grant levels.

600MON Command Set

COMMAND	DESCRIPTION
BF <address1><address2><pattern>	Block fill memory
BI <address1><address2>	Block initialize memory
BM <address1><address2><address3>	Block move memory
BR [<address>[;<count>]][<address>[;<count>]]...	Breakpoint set
NOBR [<address>[<address>]...]	Breakpoint remove
BS <address1><address2><data>[<mask>][;<option>]	Block search memory
BT <address1><address2>	Block test memory
DC <expression>	Data conversion
DF	Display registers formatted
DU [<port number>][<address1><address2>[<text>]	Dump memory as S-records
GD [<address>]	Go direct execute program
G[0][<address>]	Go execute program
GT <temp. breakpoint address>	Go until breakpoint
HE	Help
LO [<port number>][;[<options>]=<text>]	Load S-records into memory UB
MD [<port number>] <address> [<count][;DI]	Memory display/disassemble
M [M]<address>[;<options>]	Memory modify
MS <address><data>	Memory set
OF	Display offsets
PA [<port number>]	Port attach
NOPA [<port number>]	Port detach
PF [<port number>]	Port Format
R [M]	Register Modify
TM [<port number>][<exit character>[<trailing character>]]	Transparent mode
T [R] [<count>]	Trace
TT <breakpoint address>	Temp. breakpoint trace
VE [<port number>][;[<options>]=<text>]	Verify memory to S-records
.A0 - .A7 [<expression>]	Display/set address register
.D0 - .D7 [<expression>]	Display/set data register
.R0 - .R6 [<expression>]	Display/set relative offset
.PC [<expression>]	Display/set program counter
.SR [<expression>]	Display/set status register
.SS [<expression>]	Display/set s. stack pointer
.US [<expression>]	Display/set user stack pointer
(CTRL B)	Abort command
(DEL)	Delete character
(CTRL D)	Redisplay line

600MON ERROR MESSAGES

ERROR MESSAGE	MEANING
PRINTER NOT READY	Printer is not properly connected or cannot receive input
SYNTAX ERROR	Error in command line
ERROR	Error (prefix)
ILLEGAL INSTRUCTION	Instruction used an illegal op-code
.... TRAP ERROR	See Traps in MC68000 user's guide
IS NOT HEX DIGIT	Improper character entered in a field that requires hexadecimal digit
DATA DID NOT STORE	Data did not go where intended (such as attempting to write ROM)
INVALID ADDRESS	Too big (1 in bits 24 - 31) or odd for .W or .L (1 in bit 0)
WHAT	Program does not recognize user's entry
NOT HEX	Same as IS NOT HEX DIGIT
FAILED AT.. WRITE=.. READ=..	Read or write command failure output by BT
X?	Assembler syntax input is illegal, incomplete, or begins in the label field

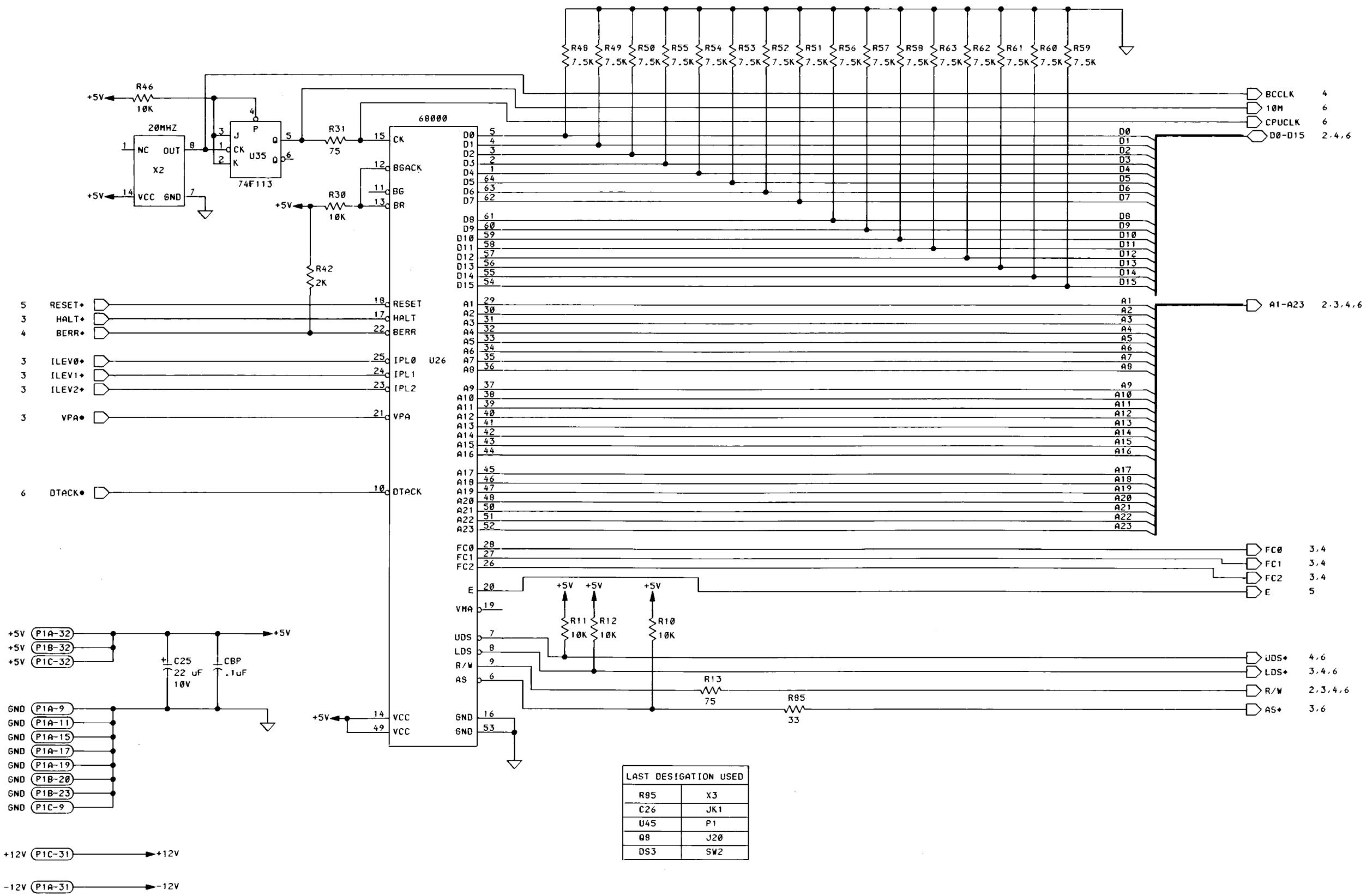
OTHER MESSAGES

VMEBUG x.y>	VMEbug prompt
HELP	HE command displays all valid commands
STOP BITS=	Displayed by PF command
CHAR NULL=	Displayed by PF command

600MON ERROR MESSAGES (cont'd)

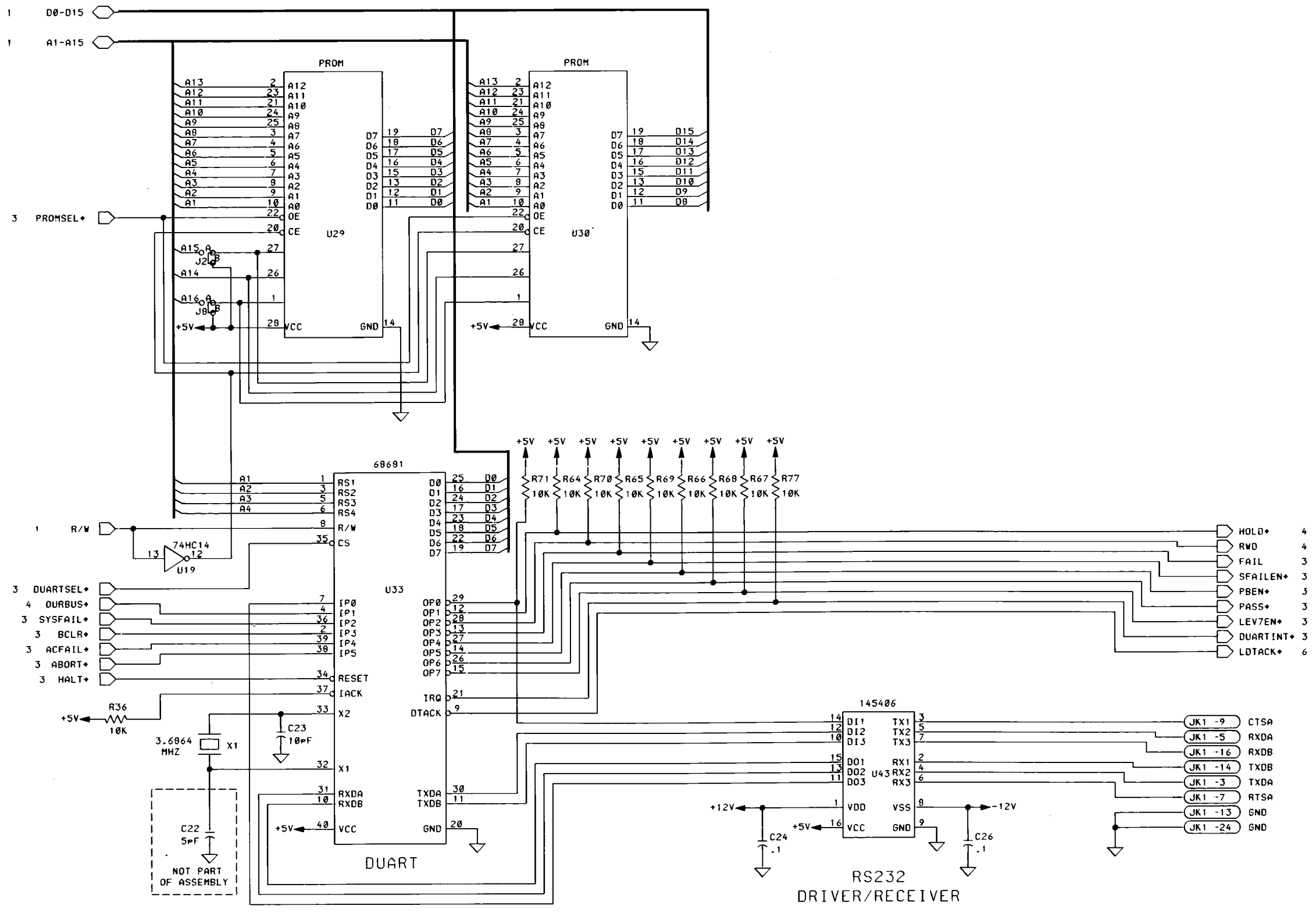
OTHER MESSAGES (cont'd)

C/R NULL	Displayed by PF command
TRANSPARENT EXIT= $01=CTL A	Displayed by TM command
SOFTWARE ABORT	Displayed when abort button is used
BREAK	BREAK key has been used
AT BREAKPOINT	Indicates program has stopped at breakpoint
PHYSICAL ADDRESS	Actual address by command
CHKSUM=	Checksum in LO or VE command
DUMP DONE	Memory dump to disk is complete
.PC within "DEBUGGER'	Displayed by trace commands

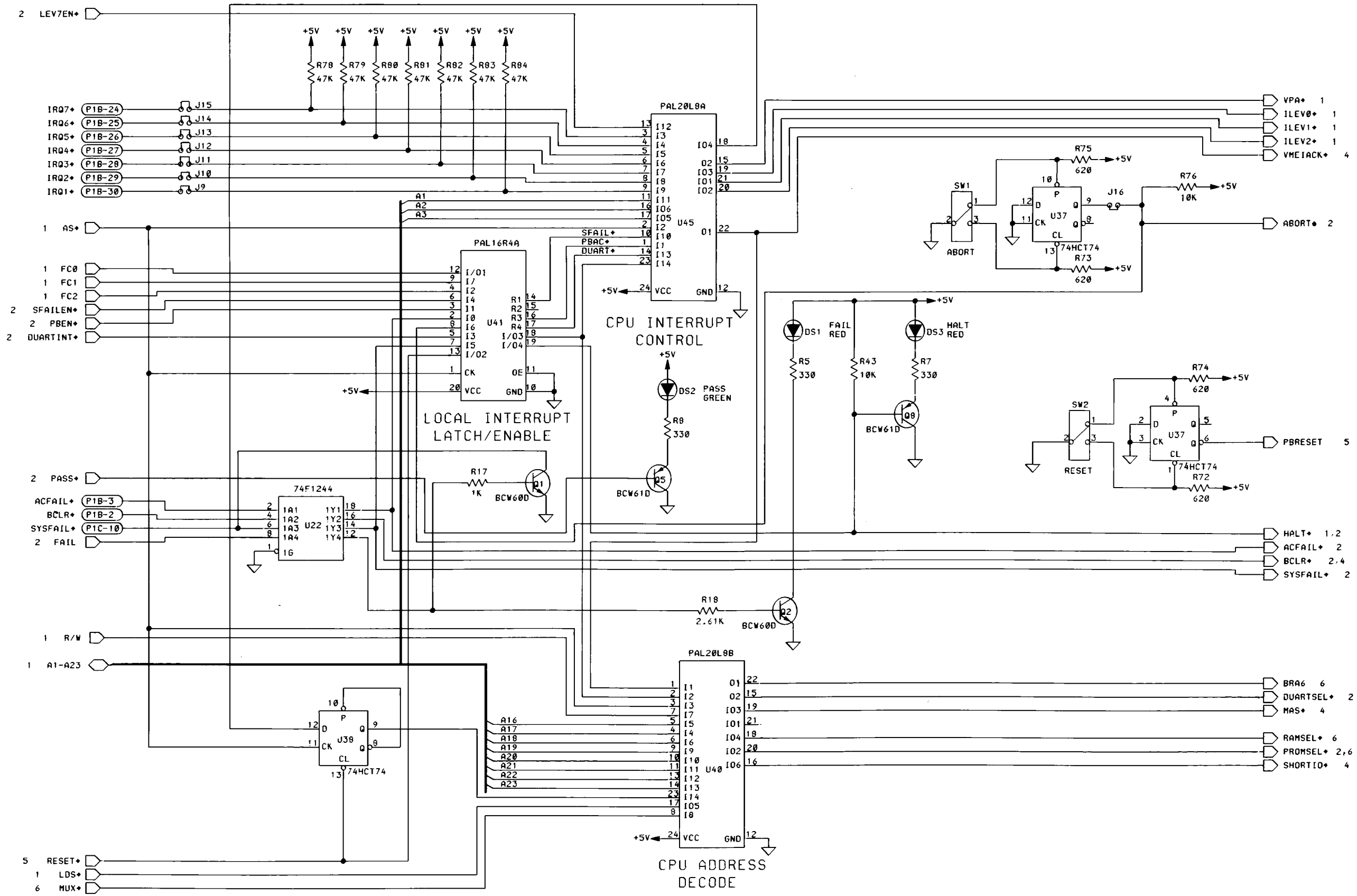


LAST DESIGATION USED	
R85	X3
C26	JK1
U45	P1
Q8	J20
DS3	SW2

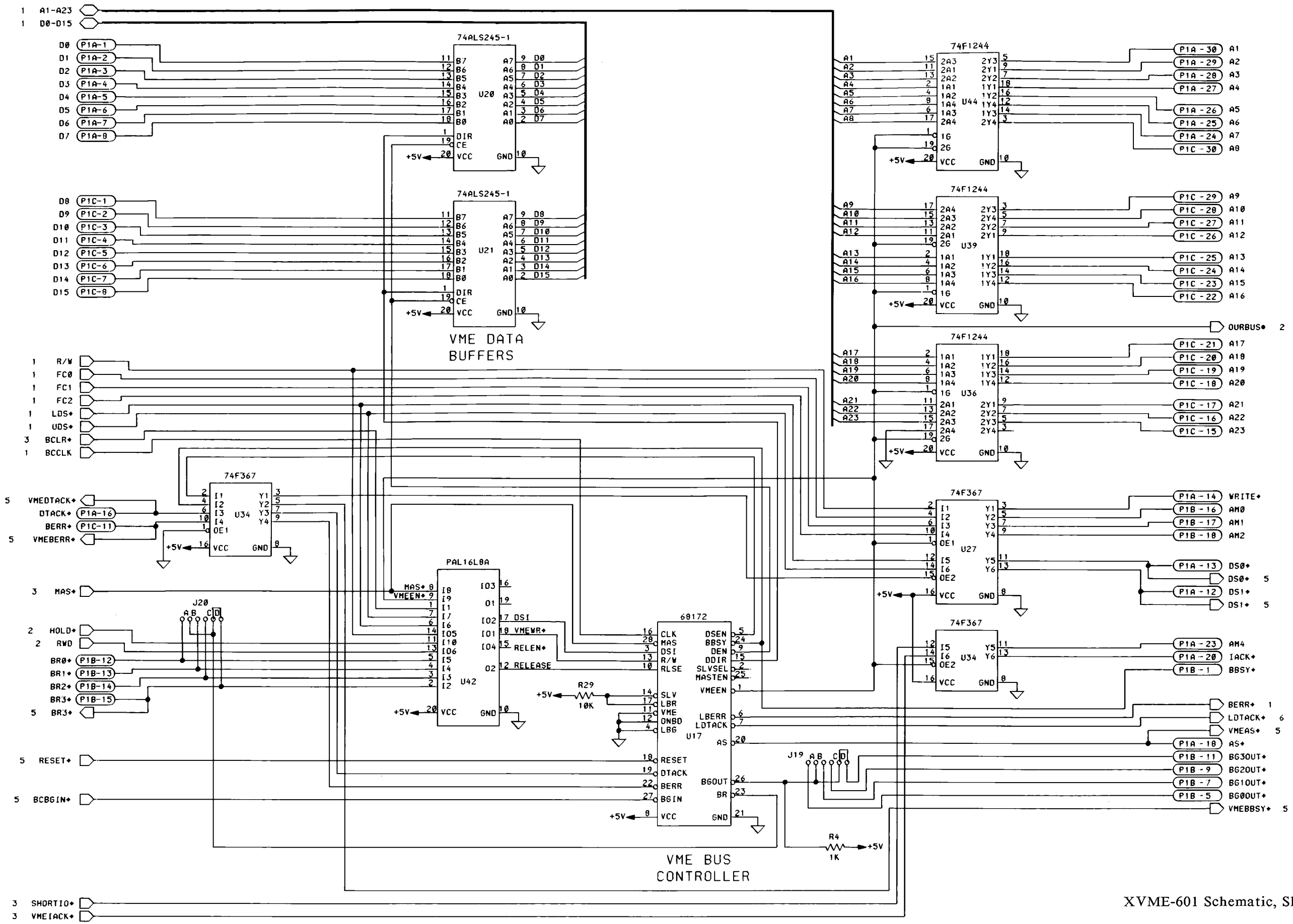
XVME-601 Schematic, Sheet 1 of 6



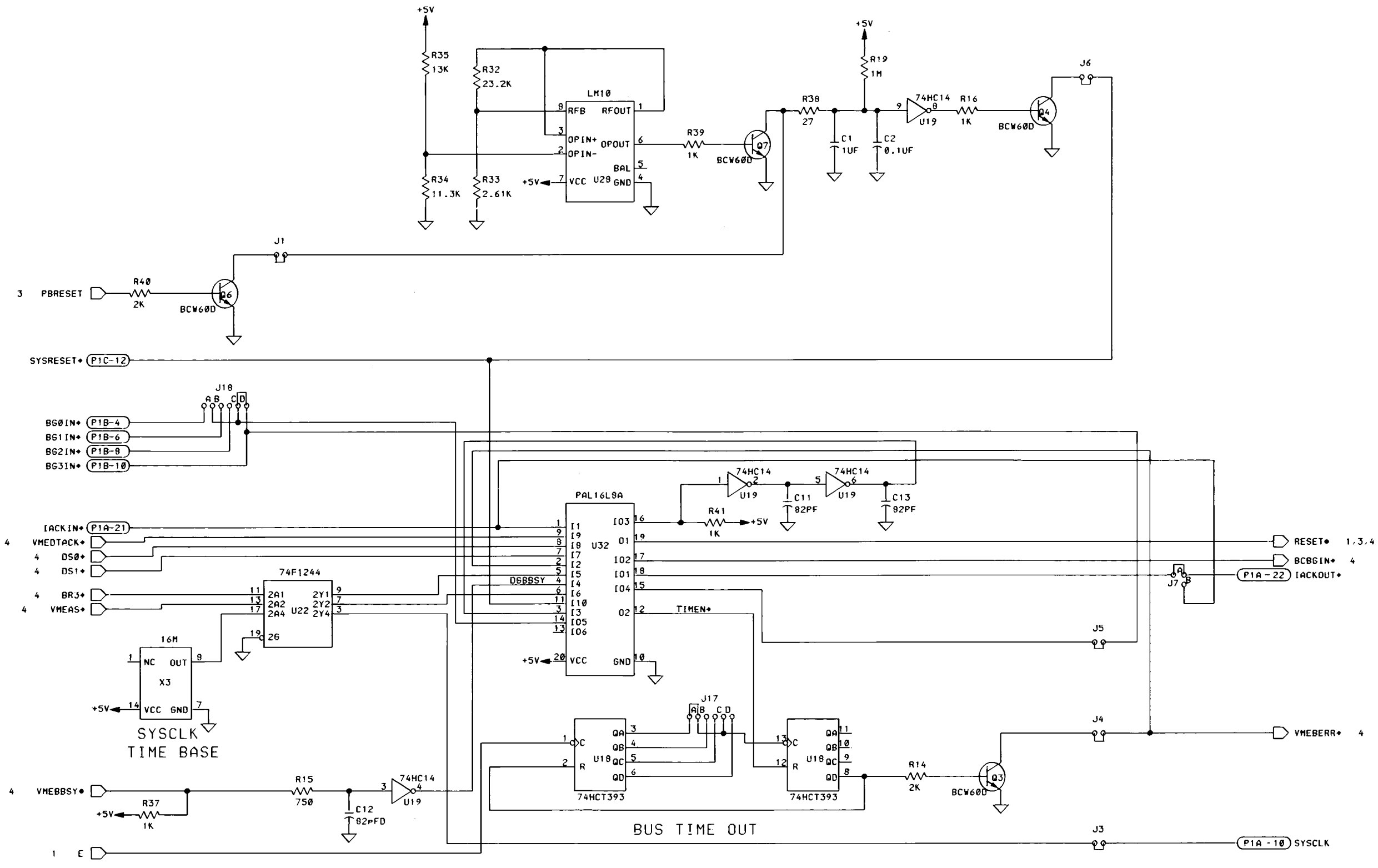
XVME-601 Schematic, Sheet 2 of 6



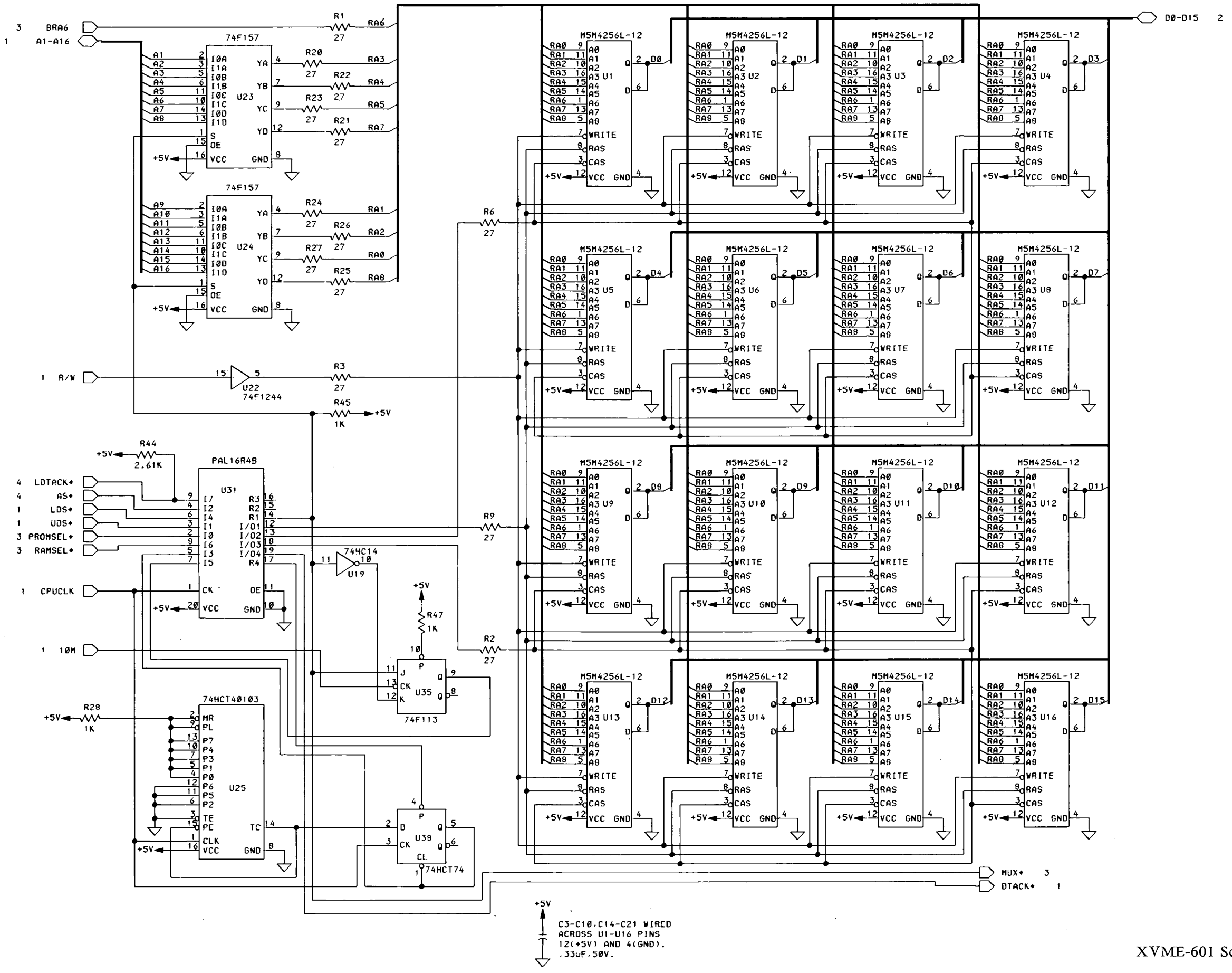
XVME-601 Schematic, Sheet 3 of 6



XVME-601 Schematic, Sheet 4 of 6



XVME-601 Schematic, Sheet 5 of 6



XVME-601 Schematic, Sheet 6 of 6